TUGAS AKHIR – TE 141599

IMPLEMENTASI *AUTONOMOUS NAVIGATION ROBOT* MENGGUNAKAN *GLOBAL POSITIONING SYSTEM* (GPS) UNTUK PEMETAAN KADAR GAS BERBAHAYA

Roby Adi Wibowo
NRP. 2213106009

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Suwito, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – TE 141599

IMPLEMENTASI *AUTONOMOUS NAVIGATION ROBOT* MENGGUNAKAN *GLOBAL POSITIONING SYSTEM* (GPS) UNTUK PEMETAAN KADAR GAS BERBAHAYA

Roby Adi Wibowo
NRP 2213106009

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Suwito, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman Ini Sengaja Dikosongkan

Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – TE 141599

IMPLEMENTATION OF AUTONOMOUS NAVIGATION ROBOT USING GLOBAL POSITIONING SYSTEM (GPS) FOR MAPPING OF HAZARDOUS GAS LEVEL

Roby Adi Wibowo
NRP 2213106009

Advisor Lecturer
Dr. Muhammad Rivai, ST., MT.
Suwito, ST., MT.

ELECTRICAL ENGINEERING DEPARTEMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman Ini Sengaja Dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Implementasi *Autonomous Navigation Robot* Menggunakan *Global Positioning System* (GPS) Untuk Pemetaan Kadar Gas Berbahaya**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri. Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2017

Roby Adi Wibowo
NRP 2213106009

Halaman Ini Sengaja Dikosongnkan

IMPLEMENTASI *AUTONOMOUS NAVIGATION ROBOT* MENGGUNAKAN *GLOBAL POSITIONING SYSTEM* (GPS) UNTUK PEMETAAN KADAR GAS BERBAHAYA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Muhammad Rivai, ST., MT.	**Suwito, ST., MT.**
NIP. 196904261994031003	**NIP. 198101052005011004**

SURABAYA
JANUARI, 2017

Halaman Ini Sengaja Dikosongkan

Implementasi *Autonomous Navigation Robot Menggunakan Global Positioning System* (GPS) Untuk Pemetaan Kadar Gas Berbahaya

Nama : **Roby Adi Wibowo**
Dosen Pembimbing 1 : **Dr. Muhammad Rivai, ST., MT.**
Dosen Pembimbing 2 : **Suwito, ST., MT**

ABSTRAK

Gas berbahaya hasil dari aktifitas alam maupun aktifitas manusia yang secara tidak sengaja muncul ke udara bebas dapat menimbulkan masalah yang serius. Keracunan massal pada suatu daerah dan ledakan pipa gas merupakan akibat dari adanya gas-gas berbahaya yang tidak terdeteksi sumber kemunculan dan penyebarannya.

Penelitian ini mengimplementasikan *Autonomous Navigation Robot* yang dilengkapi dengan *Global Positioning System* (GPS) Ublox Neo M8 dan sensor gas TGS2600 untuk melakukan aktifitas pemetaan terhadap gas berbahaya pada suatu lokasi menggunakan aplikasi JAVA berbasis Google Map. Robot *autonomous* ini akan dikirim ke tempat-tempat yang beresiko terhadap paparan gas berbahaya, bekerja dengan panduan navigasi *waypoint*. Kemudian memberikan informasi tentang kadar gas serta lokasi robot berada.

Hasil percobaan menunjukkan arah dan kecepatan angin melebihi 5 km/jam mempengaruhi kenaikan dan penurunan nilai pembacaan kadar gas, kesalahan kompas dalam memandu robot pada arah *setpoint* terbesar adalah 38%, dan kesalahan GPS dalam bernavigasi menuju *waypoint* mulai dari 2,91 meter, 3,5 meter, dan 3,81 meter.

Hasil implementasi ini menunjukkan bahwa *Autonomous Navigation Robot* yang dilengkapi dengan GPS Ublox Neo M8 dan sensor TGS 2600 dapat memetakan kadar gas suatu lokasi dengan keakuratan posisi di bawah 5 meter.

Kata Kunci: ***Autonomous Navigation Robot,*** **Gas berbahaya,** ***Global Positioning System*****, Sensor Gas TGS2600**

Halaman Ini Sengaja Dikosongkan

Implementation of Autonomous Navigation Robot Using Global Positioning System for Mapping of Hazardous Gas Level

Name ***: Roby Adi Wibowo***
1st Advisor ***: Dr. Muhammad Rivai, ST., MT.***
2nd Advisor ***: Suwito, ST., MT.***

ABSTRACT

Hazardous gases that provided from the activity of natural and human activities that inadvertently emerge into the air can cause serious problems. Mass poisoning at an area and gas pipeline explosion was caused of the presence of undetected hazardous gases source of the emergence and spread.

This study implements Autonomous Robot Navigation equipped with a Global Positioning System (GPS) Ublox Neo M8 and TGS2600 gas sensors for mapping activity of hazardous gas at a location using Google Map-based Java applications. The autonomous robot will be sent to places that are at risk of exposure to hazardous gases, working with waypoint navigation guidance. Then provide information about gas levels as well as the location of the robot is.

The experimental results indicate the direction and the wind speed exceeds 5 km / h influence increases and decreases in the value of the gas concentration readings, compass errors in guiding the robot in the direction of the largest setpoint is 38%, and an error in the GPS navigating to the waypoint ranging from 2.91 meters, 3, 5 meters and 3.81 meters.

The implementation results show that the Autonomous Robot Navigation incorporating GPS Ublox Neo M8 and TGS 2600 sensor can map a location with a gas content of precision positioning below 5 meters.

Keywords: Autonomous Navigation Robot, Global Positioning System, Hazardous Gases TGS2602 Gas Sensor

Halaman Ini Sengaja Dikosongkan

KATA PENGANTAR

Segala puji syukur penulis panjatkan atas kehadirat Allah SWT yang selalu memberikan rahmat serta hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Pada kesempatan ini penulis ingin mengucapkan ucapan terimakasih yang sebesar- besarnya kepada beberapa pihak yang telah memberikan dukungan selama proses pengerjaan tugas akhir ini, antara lain:

1. Keluarga penulis ayahanda Giyanto, ibunda Mustiah, beserta seluruh keluarga yang selalu memberikan doa, dukungan, motivasi, semangat, perhatian dan kasih sayangnya.
2. Ketua Jurusan Teknik Elektro Dr. Eng. Ardyono Priyadi, ST., M.Eng.
3. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan penelitian Tugas Akhir ini.
4. Suwito, ST., MT selaku dosen pembimbing kedua, atas masukan dan saran aplikatif yang diberikan selama pengerjaan penelitian Tugas Akhir ini.
5. Dr. Ir. Hendra Kusuma, M.Eng.Sc., Ir. Tasripan, MT., Ir. Haris Pringadi, MT., dan Fajar Pambudi, ST., M.Sc., selaku dosen penguji sidang ujian.
6. Seluruh dosen bidang studi Elektronika Jurusan Teknik Elektro FTI ITS.
7. Tira Triscahyaningrum yang selalu memberi dukungan, perhatian, dan kasih sayang selama ini.
8. Seluruh rekan-rekan yang telah banyak membantu dalam penyelesaian Tugas Akhir ini.

Penulis menyadari bahwa dalam Tugas Akhir ini terdapat banyak kekurangan. Akhir kata semoga melalui tulisan ini dapat bermanfaat dan dapat berbagi ilmu bagi pembacanya. Amin.

Surabaya, Januari 2017

Penulis

Halaman Ini Sengaja Dikosongkan

DAFTAR ISI

Halaman Ini Sengaja Dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman Ini Sengaja Dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Berdasarkan data kasus keracunan yang dilaporkan ke Sentra Informasi Keracunan Nasional sejak tahun 2010 – 2014 terdapat 51 kasus dan 13 insiden keracunan yang terjadi akibat menghirup gas berbahaya yang beracun. Selain itu terdapat beberapa kasus kebakaran dan ledakan karena gas berbahaya. Beberapa gas berbahaya yang dilaporkan diantaranya yaitu gas karbon monoksida (CO), gas karbondioksida (CO_2), gas Hidrogen Sulfida (H_2S), gas Freon, Liquid Petroleum Gas (LPG) dan gas limbah rumah sakit. Kasus keracunan dan kebakaran dari gas berbahaya yang paling sering terjadi disebabkan karena paparan gas karbon monoksida (CO), paparan gas karbondioksida (CO_2), kebocoran pipa jalur Liquid Petroleum Gas (LPG) dan bocornya atau munculnya gas Hidrogen Sulfida (H_2S) [1].

Gas-gas berbahaya dapat muncul karena faktor internal (secara alamiah) seperti debu yang beterbangan akibat tiupan angin, abu (debu) yang dikeluarkan dari letusan gunung berikut gas-gas vulkanik dan proses pembusukan sampah organik dan lain-lain. Faktor eksternal (karena hasil aktifitas manusia) seperti hasil pembakaran bahan bakar fosil, debu atau serbuk dari kagiatan industri dan pemakaian zat-zat kimia yang disemprotkan ke udara. Gas berbahaya dan beracun yang muncul kemudian dapat terdispersi ke udara dan kemudian menyebar ke lingkungan sekitarnya. Kecepatan penyebaran ini tergantung pada keadaan geografi dan meteorologi setempat [2].

Kondisi itulah yang cukup diwaspadai. Karena sifat yang berbahaya dari gas-gas tersebut, maka pada daerah dengan indikasi kemunculan dan penyebaran gas berbahaya harus diketahui lokasi dan kadarnya agar dapat meminimalkan kerugian yang akan timbul karena pencemarannya. Mengetahui penyebaran gas berbahaya pada suatu daerah tertentu dapat melalui sebuah pemetaan.

Untuk mendapatkan data sebagai dasar peta, sampel gas dapat diambil menggunakan *sampler* yang berbasis pompa. Tetapi cara ini hanya dapat mengukur konsentrasi gas di waktu dan tempat yang sama. Dengan menggunakan *Autonomous Navigation Robot* yang dilengkapi dengan *Global Positioning System* (GPS) dan sensor gas, pengambilan sampel data dapat dilakukan pada titik-titik lokasi berbeda dan lebih luas.

Oleh karna itu, pada tugas akhir ini dirancang sebuah miniatur atau *prototype* dari sebuah *Autonomous Navigation Robot* menggunakan GPS

dan sensor gas yang dilengkapi dengan aplikasi peta yang dapat memberikan suatu informasi maupun referensi kandungan gas pada titik-titik lokasi berbeda dalam bentuk peta *visual*.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan beberapa masalah. Antara lain:

1. Sensor apa yang dapat digunakan untuk mendeteksi gas berbahaya?
2. Bagaimana membuat *Autonomous Navigation Robot* menggunakan GPS?
3. Bagaimana merancang dan membuat sebuah aplikasi yang dapat menampilkan informasi dalam suatu peta?
4. Bagaimana bentuk suatu pemetaan kadar gas berbahaya pada suatu lokasi dengan mengimplementasikan *Autonomous Navigation Robot*?

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut:

1. Mampu mengetahui sensor yang dapat digunakan untuk mendeteksi gas berbahaya.
2. Mampu membuat sistem *Autonomous Navigation Robot* menggunakan GPS.
3. Mampu membuat aplikasi yang dapat menampilkan informasi ke dalam suatu peta.
4. Mampu menyajikan peta berisi informasi tampilan data lokasi dan kandungan gas berbahaya dengan mengimplementasikan *Autonavigation Robot* yang diberi sensor GPS dan sensor gas.

1.4 Batasan Masalah

1. Robot dijalankan pada tempat yang datar, tidak terdapat halangan dan tidak tergenang air.
2. Gas *iso-butane* dijadikan sebagai bahan uji gas berbahaya.
3. Robot berjalan pada keadaan kecepatan angin yang rendah dibawah 15 km/jam.
4. Robot tidak dijalankan pada radius melebihi 200 meter dari komputer *server*.
5. Komputer *server* harus terhubung dengan jaringan internet.

1.5 Metodologi Penelitian

Dalam penyelesaian tugas akhir ini digunakan metodologi sebagai berikut :

1. Studi literatur

   Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Berikut studi literatur yang dilakukan :

   a. Studi tentang protokol NMEA pada GPS UBLOX M8
   b. Studi tentang akuisisi data serial protokol NMEA.
   c. Studi tentang akuisisi data ADC pada sensor TGS2600.
   d. Studi tentang pemrograman komputer berbasis JAVA.

Dasar teori ini dapat diambil dari buku-buku, jurnal, proceeding, dan artikel-artikel di internet.

2. Perancangan Sistem

   Setelah mempelajari literatur yang ada, selanjutnya akan dilakukan perancangan sistem. Sistem yang akan dirancang meliputi integrasi modul-modul sensor, *driver* motor, dan telemetri.

3. Pengujian Sistem

   Pengujian dilakukan pada *software* dan *hardware* yang telah dibuat untuk menguji keakuratan alat dengan mencoba menjalankannya pada lapangan terbuka, disertai memberikan beberapa lokasi titik yang diberi gas.

4. Pengolahan Data

   Data yang telah diperoleh dari komputer *server*, modul sensor kompas, sensor GPS, dan sensor gas diakuisisi oleh mikrokontroler pada minimum sistem. Data dari komputer *server* diolah untuk digunakan sebagai acuan tujuan, data dari sensor kompas dan GPS diolah sebagai navigasi menuju tujuan, dan data dari sensor gas diolah untuk dapat mengidentifikasi nilai gas.

5. Penulisan Laporan Tugas Akhir

   Tahap penulisan laporan Tugas Akhir dilakukan pada saat pengujian sistem dimulai dan setelahnya.

1.6 Sistematika Penulisan

Laporan Tugas Akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

- Bab 1: Pendahuluan

  Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan, dan relevansi.

- Bab 2: Dasar Teori

  Bab ini menjelaskan tentang dasar-dasar teori tentang gas berbahaya, sensor gas TGS2600, kompas digital, GPS, protokol NMEA yang digunakan untuk GPS, sistem navigasi, papan mikrokontroler, modul komunikasi data antara robot dengan komputer, dan pembuatan aplikasi peta berbasis JAVA.

- Bab 3: Perancangan Alat

  Bab ini menjelaskan tentang perancangan perangkat keras dan perangkat lunak untuk pengaplikasian penelitian.

- Bab 4: Pengujian Alat

  Bab ini menjelaskan tentang hasil yang didapat dalam pengujian dari tiap blok sistem dan subsistem serta hasil evaluasi sistem tersebut.

- Bab 5: Penutup

  Bab ini menjelaskan tentang kesimpulan meliputi hasil yang dicapai, dan kekurangan-kekurangan pada kerja alat dari hasil analisa serta saran untuk pengembangan ke depan.

1.7 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut :

1. Dapat mendukung penelitian khususnya pada bidang studi elektronika tentang penggunaan sensor gas dalam aplikasi pengukuran dan penelitian selanjutnya.
2. Mendukung penelitian tentang robotika dengan navigasi otomatis yang dapat dimanfaatkan untuk keperluan lebih luas.
3. Dihasilkan aplikasi komputer untuk mendukung penelitian tentang pemonitoran, pendeteksian, dan pemetaan kadar gas pada ruang terbuka.

BAB II
TEORI PENUNJANG

2.1. Gas Beracun dan Berbahaya

Terdapat beberapa macam gas pengotor dalam udara bebas atau tambang bawah tanah dapat digolongkan menjadi gas beracun dan gas berbahaya. Gas-gas ini berasal dari proses-proses yang terjadi dalam tambang, batuan ataupun bahan galiannya, maupun berasal dari proses aktifitas manusia. Gas-gas yang bersifat gas beracun adalah gas yang bereaksi dengan darah dan dapat menyebabkan kematian. Sedangkan gas berbahaya adalah gas gas pengotor yang menyebabkan bahaya, baik terhadap kehidupan manusia maupun dapat menyebabkan peledakan. Beberapa gas yang tergolong dalam gas beracun adalah:

a. Karbondioksida (CO_2)

Gas ini tidak berwarna dan tidak berbau dan tidak mendukung nyala api dan bukan merupakan gaa mudah terbakar. Gas ini lebih berat dari pada udara, karenanya selalu terdapat pada bagian bawah dari suatu jalan udara. Dalam udara normal kandungan CO_2 adalah 0,03%. Dalam tambang bawah tanah sering terkumpul pada bagian bekas-bekas penambangan terutama yang tidak terkena aliran ventilasi, juga pada dasar sumur-sumur tua. Sumber dari CO_2berasal dari hasil pembakaran, hasil peledakan atau dari lapisan batuan dan dari hasil pernapasan manusia. Pada kandungan CO_2 = 0,5% laju pernapasan manusia mulai meningkat, pada kandungan CO_2 = 3% laju pernapasan menjadi dua kali lipat dari keadaan normal, dan pada kandungan CO_2 = 5% laju pernapasan meningkat tiga kali lipat dan pada CO_2 = 10% manusia hanya dapat bertahan beberapa menit. Kombinasi CO_2 dan udara biasa disebut dengan ‘blackdamp’.

b. Karbon Monoksida (CO)

Gas karbon monoksida merupakan gas yang tidak berwarna, tidak berbau dan tidak ada rasa, dapat terbakar dan sangat beracun. Gas ini banyak dihasilkan dari sisa pembakaran senyawa hidrokarbon. Gas ini mempunyai afinitas yang tinggi terhadap haemoglobin darah, sehingga sedikit saja kandungan gas CO dalam udara akan segera bersenyawa dengan butir-butir haemoglobin (COHb) yang akan meracuni tubuh lewat darah. Afinitas CO terhadap haemoglobin menurut penelitian (Forbes and Grove, 1954) mempunyai kekuatan 300 kali lebih besar dari pada oksigen

dengan hemoglobin. Gas CO dihasilkan dari hasil pembakaran, operasi motor bakar, proses peledakan dan oksidasi lapisan batubara.

Karbon monoksida merupakan gas beracun yang sangat mematikan karena sifatnya yang kumulatif. Misalnya gas CO pada kandungan 0,04% dalam udara apabila terhirup selama satu jam baru memberikan sedikit perasaan tidak enak, namun dalam waktu 2 jam dapat menyebabkan rasa pusing dan setelah 3 jam akan menyebabkan pingsan/ tidak sadarkan diri dan pada waktu lewat 5 jam dapat menyebabkan kematian. Kandungan CO sering juga dinyatakan dalam ppm (part per milion). Sumber CO yang sering menyebabkan kematian adalah gas buangan dari mobil dan kadang-kadang juga gas pemanas air. Gas CO mempunyai berat jenis 0,9672 sehingga selalu terapung dalam udara.

c. Hidrogen Sulfida (H_2S)

Gas ini sering disebut juga '*stinkdamp*' karena baunya seperti bau telur busuk. Gas ini tidak berwarna, merupkan gas racun dan dapat meledak, merupakan hasil dekomposisi dari senyawa belerang. Gas ini mempunyai berat jenis yang sedikit lebih berat dari udara. Merupakan gas yang sangat beracun dengan ambang batas *Threshold Limit Value - Time Weighted Average* (TLV-TWA) sebesar 10 ppm pada waktu selama 8 jam terpapar dan untuk waktu singkat *Threshold Limit Value - Short-term Exposure Limit* (TLV-STEL) adalah 15 ppm. Walaupun gas H_2S mempunyai bau yang sangat jelas, namun kepekaan terhadap bau ini akan dapat rusak akibat reaksi gas H_2S terhadap saraf penciuman. Pada kandungan H_2S = 0,01 % untuk selama waktu 15 menit, maka kepekaan manusia akan bau ini sudah akan hilang.

d. Sulfur Dioksida (SO_2)

Sulfur dioksida merupakan gas yang tidak berwarna dan tidak bisa terbakar. Merupakan gas racun yag terjadi apabila ada senyawa belerang yang terbakar. Lebih berat dari pada udara, dan akan sangat membantu pada mata, hidung dan tenggorokan. Harga ambang batas ditetapkan pada keadaan gas = 2 ppm (TLV-TWA) atau pada waktu terdedah yang singkat (TLV-STEL) = 5 ppm.

e. Nitrogen Oksida (NO_x)

Gas nitrogen oksida sebenarnya merupakan gas yang '*inert*', namun pada keadaan tekanan tertentu dapat teroksidasi dan dapat menghasilkan gas yang sangat beracun. Terbentuknya dalam tambang bawah tanah sebagai hasil peledakan dan gas buang dari motor bakar. NO_2 merupakan

gas yang lebih sering terdapat dalam tambang dan merupakan gas racun. Harga ambang batas ditetapkan 5 ppm, baik untuk waktu terdedah singkat maupun untuk waktu 8 jam kerja. Oksida nitrogen yang merupakan gas racun ini akan bersenyawa dengan kandungan air dalam udara membentuk asam nitrat, yang dapat merusak paru-paru apabila terhirup oleh manusia.

Yang tergolong dalam gas berbahaya yaitu:

a. Metana (CH_4)

Gas metana ini merupakan gas yang selalu berada dalam tambang batubara dan sering merupakan sumber dari suatu peledakan tambang. Campuran gas metana dengan udara disebut '*Firedamp*'. Apabila kandungan metana dalam udara tambang bawah tanah mencapai 1% maka seluruh hubungan mesin listrik harus dimatikan. Gas ini mempunyai berat jenis yang lebih kecil dari pada udara dan karenanya selalu berada pada bagian atas dari jalan udara. Metana merupakan gas yang tidak beracun, tidak berwarna, tidak berbau dan tidak mempunyai rasa. Pada saat proses pembatubaraan terjadi maka gas metana terbentuk bersama-sama dengan gas karbondioksida. Gas metana ini akan tetap berada dalam lapisan batubara selama tidak ada perubahan tekanan padanya.

Terbebasnya gas metana dari suatu lapisan batubara dapat dinyatakan dalam suatu volume per satuan luas lapisan batubara, tetapi dapat juga dinyatakan dalam satuan volume per satuan waktu. Terhadap kandungan gas metana yang masih terperangkap dalam suatu lapisan batubara dapat dilakukan penyedotan dari gas metana tersebut dengan pompa untuk dimanfaatkan. Proyek ini dikenal dengan nama '*seam methane drainage*'.

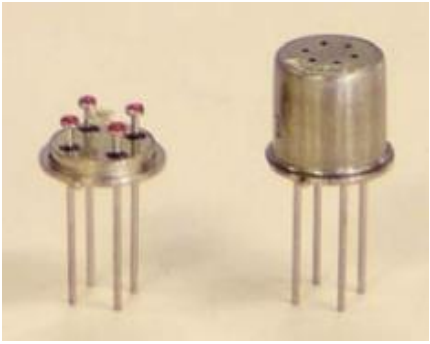
b. Liquified Petroleum Gas (LPG)

Liquified Petroleum Gas (LPG), merupakan gas minyak bumi yang dicairkan. Dengan menambah tekanan dan menurunkan suhunya, gas berubah menjadi cair. Komponennya didominasi propana (C_3H_8) dan butana (C_4H_{10}). Elpiji juga mengandung hidrokarbon ringan lain dalam jumlah kecil, misalnya etana (C_2H_6) dan pentana (C_5H_{12}). Dalam kondisi atmosfer, elpiji akan berbentuk gas. Volume elpiji dalam bentuk cair lebih kecil dibandingkan dalam bentuk gas untuk berat yang sama. Gas ini

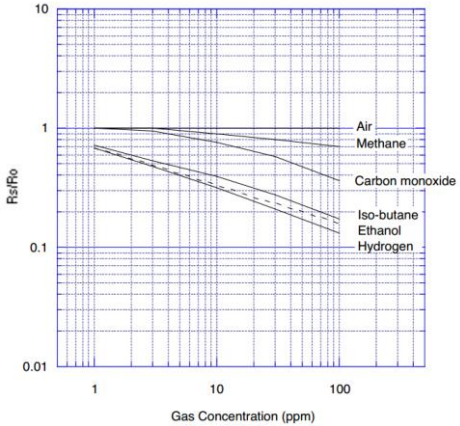
mempunyai sifat sangat mudah terbakar, mempunyai masa yang lebih berat dari udara, tidak beracun, tidak berwarna, tetapi berbau menyengat.

2.2. Sensor Gas TSG2600

Elemen sesnor terdiri dari lapisan semikonduktor *metal-oxide* yang dibentuk pada subtrat alumina sensor yang terinterasi dan digabung dengan pemanas. Ketika gas polutan terdeteksi, maka konduktivitas sensor meningkat tergantung pada kandungan atau kadar gas di udara. Sebuah rangkaian listrik yang terdiri dari resistor, dapat mengkonversi perubahan konduktivitas untuk sinyal keluaran yang sesuai dengan konsentrasi gas. TGS 2600 (Gambar 2.1) memiliki sensitivitas yang tinggi untuk konsentrasi gas rendah dari kontaminan udara seperti hidrogen dan karbon monoksida yang ada dalam asap rokok. Sensor dapat mendeteksi hidrogen pada tingkat beberapa ppm. TGS 2600 membutuhkan arus pemanas hanya 42mA. Pada Gambar 2.2 menunjukkan karakteristik sensitivitas secara umum. Semua data yang telah terkumpul dalam tes uji coba, memperlihatkan sumbu Y merupakan perbandingan resistansi (Rs/Ro) di mana Rs merupakan resistansi sensor ketika mendeteksi gas, dan Ro merupakan resistansi sensor pada udara bersih [3].



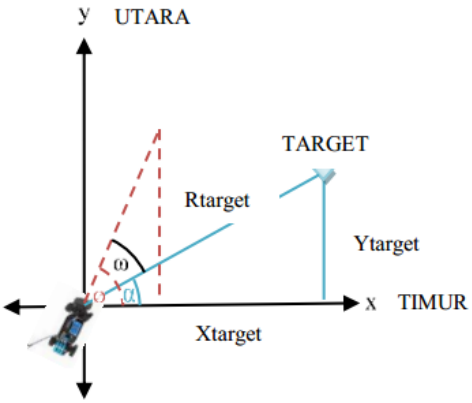
Gambar 2.1 Sensor Gas Figaro TGS2600 [3]



Gambar 2.2 Karakteristik sensor gas Figaro TGS2600 [3]

2.3. Navigasi *Waypoint*

Autonavigation robot merupakan kemampuan robot dalam berjalan secara otomatis menuju suatu titik tujuan pada bidang x-y secara otomatis tanpa ada campur tangan manusia [4].



Gambar 2.3 Penggambaran titik target terhadap objek [4]

Metode umum untuk mewujudkan navigasi otomatis ini dapat melalui navigasi visual, model peta navigasi, dan navigasi *waypoint*. *Waypoint* merupakan kumpulan dari koordinat yang mengenalkan sebuah titik pada ruang fisik.

Gambar 2.4 Titik navigasi waypoint [5]

Koordinat yang digunakan dapat bervariasi tergantung pada aplikasi. Misalnya yang berkenaan dengan navigasi pada permukaan bumi, koordinat ini dapat mencakup *latitude* dan *longitude*. Pada navigasi udara, dikenal juga dengan *altitude* [5].

Waypoint menjadi memiliki cakupan yang luas dalam bidang navigasi oleh orang awam, semenjak dikembangkannya sistem navigasi lanjutan. *Waypoint* yang ditempatkan di atas permukaan bumi, biasanya didefinisikan dalam dua dimensi (contoh *latitude* dan *longitude*).

Pada navigasi *waypoint* dalam cakupan dua dimensi, penentuan posisi target terhadap suatu benda objek, dapat dilihat pada Gambar 2.3. Pada Gambar 2.4 dapat dilihat bahwa jarak antara posisi benda dengan target dapat dihitung dengan menggunakan persamaan 2.3

$$r_{target} = \sqrt{{x_{target}}^2 + {y_{target}}^2} \tag{2.3}$$

Variabel Xtarget dan Ytarget merupakan koordinat *latitude* dan *longitude* yang akan dituju. Sehingga arah yang harus dituju oleh benda (ω) didapat dari persamaan 2.5

$$\alpha = tan^{-1}\left(\frac{y_{target}}{x_{target}}\right) \tag{2.4}$$

$$\omega = \alpha - \emptyset \tag{2.5}$$

Sudut α pada persamaan 2.4 merupakan sudut yang dibentuk setelah menentukan titik target, dan sudut Ø merupakan sudut yang didapat dari kompas.

2.3.1. Global Positioning System *(GPS)*

GPS adalah sistem navigasi dan penentuan posisi berbasis satelit yang dapat digunakan oleh pengguna dengan memanfaatkan sinkronisasi satelit, serta didesain untuk memberikan posisi dan kecepatan tiga-dimensi yang teliti, dan juga informasi waktu secara kontinyu di seluruh dunia

Satelit GPS mengelilingi bumi dengan orbit masing-masing seperti ditunjukkan pada gambar 2.5, dan merupakan instrument yang luar biasa. Di mana masing-masing satelit memiliki empat jam atomik yang bekerja pada level satu detik error dalam tiga juta tahun. Satelit GPS mengirim deretan kode berdasarkan sinyal waktu yang datang. Tidak dari posisi atau pergerakan [6][7].

Format data keluaran GPS ditetapkan oleh *National Maritime Electronic Association* (NMEA) ada lima jenis, yaitu NMEA 0180, NMEA 0812, NMEA 0813, AVIATON, dan PLOTTING. Format data tersebut dapat dibaca oleh komputer melalui komunikasi *serial*. Data keluaran dalam format NMEA 0183 berupa kalimat (*string*) yang merupakan karakter ASCII 8 bit. Setiap awal kalimat diawali dengan karakter ”$”, dua karakter *Talker* ID, tiga karakter *Sentence* ID, dan diikuti oleh data fields yang masing-masing dipisahkan oleh koma (,) serta diakhiri oleh *optional checksum* dan karakter *carriage return/line feed* (CR/LF).

Gambar 2.5 Jalur orbit satelit [6]

Sebuah GPS mempunyai *Talker* ID berupa GP. Jenis kalimat yang dihasilkan ada beberapa macam, diantaranya adalah kalimat *Recommended Minimum Specific* (RMC), dan *Global Positioning Fix Data* (GGA) dapat dilihat pada tabel 2.1. Bentuk format umum untuk data keluaran dari GPS sebagai berikut:

$<Alamat>,<Data>*<Checksum>,<CR><LF>

$	: Karakter Awal
<alamat>	: Bagian alamat. "aa" adalah *talker identifier*, "ccc" identitas tipe.
,	: Bagian *delimiter* atau pembatas
<Data>	: Blok data
*	: *Checksum Delimiter*
<Checksum>	: Bagian Checksum
<CR><LF>	: Akhir dari barisan data (*Carriage Return, Line Feed*)

Tabel 2. 1 Format kalimat ID data GPS

$GNGGA	Waktu, Posisi, dan Perbaikan data yang terkait penerima
$GNGLL	Waktu, posisi, dan status perbaikan
$GNGSA	Digunakan untuk merepresentasikan posisi yang tetap. Ketika kedua satelit GPS dan Beidou
$GPGSA	digunakan dalam solusi posisi. Kalimat $GNGSA digunakan untuk satelit GPS, dan $GNGSA yang lain digunakan untuk satelit Beidou.
$BDGSA	Ketika hanya satelit GPS yang digunakan untuk penetapan posisi, $GPGSA sebagai keluaran. Jika satelit Beidou, maka $BDGSA sebagai keluaran
$GPGSV	Informasi dari satelit terkait *azimuth,* elevasi, dan CNR. $GPGSA digunakan untuk satelit GPS
$BDGSV	dan $BDGSV untuk satelit Beidou
$GNRMC	Waktu, tanggal, dan kecepatan perjalanan
$GNVTG	Perjalanan dan kecepatan relatif terhadap tanah
$GNZDA	UTC, hati, tanggal, dan tahun

Format NMEA juga mendukung pesan lainnya, yang pertama *Global Positioning Fix Data* (GGA) waktu, posisi, dan tetapan penerima data GPS. Format ini dijelaskan pada tabel 2.2. Yang kedua adalah *Recommended Minimum Specifik GNSS Data* (RMC). Data waktu, tanggal, posisi, perjalanan dan kecepatan diperoleh dari penerima GNSS. Format ini dijelaskan pada tabel 2.3.

Format data GGA sebagai berikut:

$--GGA,hhmmss.ss,llll.lll,a,yyyyy.yy,a,x,uu,xxxxxx,,,v*hh<CR><LF>

Tabel 2. 2 Format pesan GGA

Bagian	Nama	Deskripsi
hhmmss.ss	Waktu UTC	Posisi UTS dalam format hhmmss.ss(000000.00 ~235959.99)
llll.lll	Latitude	Format latitud dalam ddmm.mmmm(Derajat, menit.menit)
A	Indikator	N = Utara, S = Selatan
yyyyy.yyy	Longnitude	Format longitude dddmm.mmmm(Derajat, menit.menit)
A	Indikator	E = Timur, W = Barat
x	Indikator kualitas GPS	0 : Belum terkunci tepat
		1 : Posisi valid, mode SPS
		2 : Posisi valid, mode DGPS
uu	Penggunaan satelit	Jumlah satelit yang digunakan(0~24)
v.v	HDOP	Horizontal Delution of Precision(00.0~99.9)
w.w	Altitude	Ketinggian dari permukaan laut
x.x	Geoidal Separation	Meter
zzzz	ID Stasiun DGPS	0000~1023, null jika tidak ada
hh	Checksum	

Format data RMC sebagai berikut:

$--RMC,hhmmss.ss,x,llllll.ll,a,yyyyy.yy,a,x.x,u.u,xxxxxx,,,v*hh<CR><LF>

Tabel 2. 3 Format pesan RMC

Bagian	Nama	Deskripsi
hhmmss.ss	Waktu UTC	Waktu UTC dengan format hhmmss.ss(000000.00~235959.99)
x	Status	V = Peringatan penerima navigasi,
		A = Data Valid
llll.lll	Latitude	Format latitud dalam ddmm.mmmm(Derajat, menit.menit)
a	Indikator	N = Utara, S = Selatan
yyyyy.yyy	Longitude	Format longitude dddmm.mmmm(Derajat, menit.menit)
a	Indikator	E = Timur, W = Barat
x.x	Kecepatan Permukaan	Knot(000.0~999.9)
u.u	Perjalanan Permukaan	Sudut perjalanan(000.00~359.99)
xxxxxx	Tanggal UTC	Posisi tanggal UTC ddmmyy
v	Indikator Mode	N = Data tidak valid
		A = Mode Autonomous
		D = Mode Diferensial
		E = Estimasi
hh	Checksum	

2.3.2. *Global Positioning System* (GPS) Ublox Neo M8

Ublox M8 merupakan modul GPS dengan satelit GNSS dan memiliki fitur lengkap yang tersedia dalam standar industri, mudah untuk

diintegrasikan dan dikombinasikan dengan fleksibilitas daya, desain dan pilihan. Bentuk fisik dari modul GPS ini ditunjukkan pada Gambar 2.6. Sumber catu daya Neo M8 dapat menggunakan 3,3 volt atau dapat menggunakan 5 volt.

Modul GPS Neo M8 ini menggunakan komunikasi *Universal Asynchronous Receiver Transmitter* (UART) Transistor-Transistor Logic (TTL) Rx/Tx. Dengan *baudrate* awal 4800 dan dapat diatur dalam konfigurasi. Pada jalur komunikasi Rx/Tx GPS Neo M8 perlu diberi pengaman, untuk menghindari interferensi elektromagnet. Karena jalur sinyal I/O dengan panjang melebihi 3 mm dapat menjadi sebuah antena dan mungkin membawa sinyal radio frekuensi menjadi derau pada GPS. Untuk menghalau interferensi sinyal, direkomendasikan menggunakan resistor > 20 Ω, *ferrite beads* seperti BLM15HD102SN1, atau induktor seperti LQG15HS47NJ02 yang dipasang secara seri pada jalur I/O seperti yang ditunjukkan pada Gambar 2.7. Spesifikasi dari modul GPS Ublox M8 ini ditunjukkan pada Tabel 2.4.

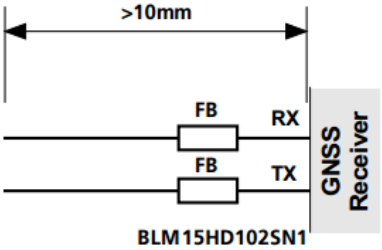
Konfigurasi dari pin pada Gambar 2.6 adalah sebagai berikut:

VCC : Pin supply +5 Volt
Rx : Pin data serial penerima (*Receiver*).
Tx : Pin data serial pengirim (*Transmitter*)
GND : Pin supply 0 Volt.

Gambar 2.6 Modul GPS Ublox Neo-M8 [8]

Tabel 2. 4 Spesifikasi GPS Ublox Neo M8

Parameter	Spesifikasi
Receiver type	72 *Channels* GPS L1 *frequency*, C/A Code SBAS: WAAS, EGNOS, MSAS, GLONASS, BEIDOU, QZSS
Time-To-First-Fix	*Cold start* : 26 s *Warm start* : 4 s *Hot Start* : 1 s
Sensitivity	*Tracking & Navigation* : -167 dBm *Reacquisition* : -160 dBm *Cold Start (without aiding)* : -148 dBm *Hot Start* : -156 dBm
Maximum Navigation update rate	5Hz
Horizontal position accuracy	GPS : 2.5 m SBAS : 2.0 m
Heading accuracy	0.3 *degrees*
Operational Limits	*Dynamics* : ≤ 4 g *Altitude* : 50,000 m *Velocity* : 500 m/s



Gambar 2.7 Perlindungan terhadap interferensi radio frekuensi [9]

Format protokol data yang digunakan oleh modul ini adalah NMEA0183. Pada protokol NMEA, data *latitude* dan *longitude* menggunakan format sumbu derajat, menit, dan detik. Untuk itu diperlukan konversi data dari nilai derajat menjadi radian yang ditunjukkan pada persamaan 2.6 hingga 2.7.
Konversi dari ddmm.mmm ke dd.dddd:

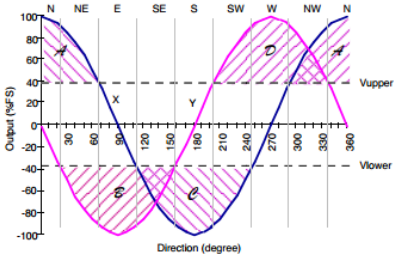
$$0.ddd = mm.mmmm/60 \quad (2.6)$$
$$dd.dddd = dd + 0.dddd \quad (2.7)$$

2.3.3. Heading

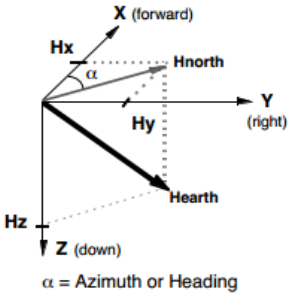
Ada dua bentuk metode kompas yang digunakan dalam sistem navigasi yang menggunakan sensor magnetoresistif adalah *the eight point compass* dan *the one-degree compass*. *Eight Point Compass* merupakan metode *heading* menggunakan penggambaran sederhana delapan titik kardinal kompas (U, S, B, T) dan titik tengah (TL, TG, BD, BL) [10].

Tipe kompas ini banyak digunakan dalam otomotif sederhana di mana pengendara ingin mengetahui arah perjalanan secara umum. Untuk aplikasi ini, sensor magnetik dapat direduksi menjadi hanya dua sumbu. Hanya menggunakan sumbu X, dan Y. Untuk menganalisa respon sensor magnetoresistif, gambar keluaran X, dan Y saat berjalan memutar yang ditunjukkan pada Gambar 2.8. Dengan mengetahui medan magnet bumi selalu menunjuk arah utara, mulai analisa dengan sumbu X (perjalanan) lurus ke utara. Keluaran sumbu X akan berada pada nilai maksimumnya ketika keluaran Y berada pada 0. Saat perjalan belok searah jarum jam menuju arah timur, sumbu X akan berkurang nilainya hingga 0, dan sumbu Y akan berkurang hingga nilai keluaran pada negatif maksimal.

Lengkungan X dan Y pada Gambar 2.8 dapat disusun menjadi delapan daerah yang merepresentasikan empat titik kardinal dan titik tengah kompas.



Gambar 2.8 Sumbu keluaran arah metode *eight point compass* [10]



Gambar 2.9 Vektor sumbu kompas [10]

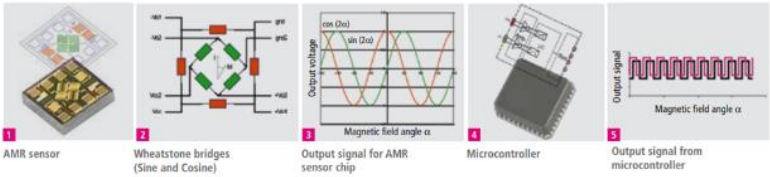
Yang kedua adalah *One Degree Compass.*Yaitu metode *heading* yang mencapai ketelitian 1º. Kompas membutuhkan sensor magnet yang dapat secara nyata memperbaiki perubahan *angular* hingga 0.1º. Sensor juga harus menunjukkan histerisis yang rendah (<0.05%FS), linearitas derajat yang tinggi (<0.5%FS *error*) dan dapat dipakai berulang-ulang (konsisten). Medan magnet pada bidang X dan Y biasanya dapat mencapai sekitar 200 hingga 300 miligauss pada bagian ekuator dan kurang pada bagian sumbu. Sehingga berlaku hubungan pada persamaan 2.9. Secara vektor ditunjukkan pada Gambar 2.9.

$$Azimuth = tan^{-1}\frac{y}{x} \tag{2.9}$$

2.3.4. *Kompas Digital HMC5883L*

HMC5883L merupakan jenis sensor magnetoresistif yang dapat mengukur medan magnet hingga mencapai 2 mili-Gauss pada resolusi medan +/- 8 Gauss. Saat sumber daya dipasang, sensor akan merubah tiap pengaruh medan magnet dalam arah sumbu yang peka menjadi perbedaan tegangan. Sensor magnetoresistif dibuat dari bahan film tipis *nickel-iron* (*Permalloy*) yang bermotif sebagai elemen jalur resistif. Adanya medan magnet mengubah elemen jembatan resistif yang menyebabkan perubahan tegangan sesuai tegangan yang melintasi keluaran jembatan [11].

Sumber arus internal menghasilkan arus DC sebesar 10 mA dari sumber VDD. Arus DC ini yang diterapkan menjadi ikatan seimbang pada sensor magnetoresistif, yang mana menghasilkan bias medan magnet buatan sensor.



Gambar 2.10 Proses menghasilkan data sumbu HMC5883L [11]

Perbedaan pengukuran dan keadaan medan ini diambil untuk menjadi tiga sumbu register keluaran X, Y, dan Z. Proses ini ditunjukkan pada Gambar 2.10. Spesifikasi dari sensor ini ditunjukkan pada Tabel 2.5.

Tabel 2. 5 Spesifikasi Sensor Kompas HMC5883L

Karakteristik	Kondisi	Min	Typ	Max	Unit
Supply	VDD Referensi pada AGND	2.6	2.5	3.6	Volts
	VDD Refensi pada DGND	1.71	1.8	VDD+0.1	Volts
Arus Rata-rata	Idle	-	2	-	µA
	Pengukuran	-	100	-	µA
Jangkauan Medan	Skala penuh	-8		8	gauss
Sensitivitas	VDD=3V, GN=0 sampai 7, 12-bit ADC	230		1370	LSb/gauss
Resolusi	VDD=3V, GN=0 sampai 7, 12-bit ADC	0.73		4.35	mili-gauss
Derau dasar			2		mili-gauss
Liniearitas	2.0 gauss jangkauan masukan			0.1	%FS
Alamat I2C	8-bit alamat baca		0x3D		hex
	8-bit alamat tulis		0x3C		hex
I2C rate				400	kHz

Pin dan bentuk fisik dari modul kompas digital HMC5883L ini ditunjukkan pada Gambar 2.11

Konfigurasi pin dari kompas digital dari Gambar 2.11 adalah sebagai berikut:

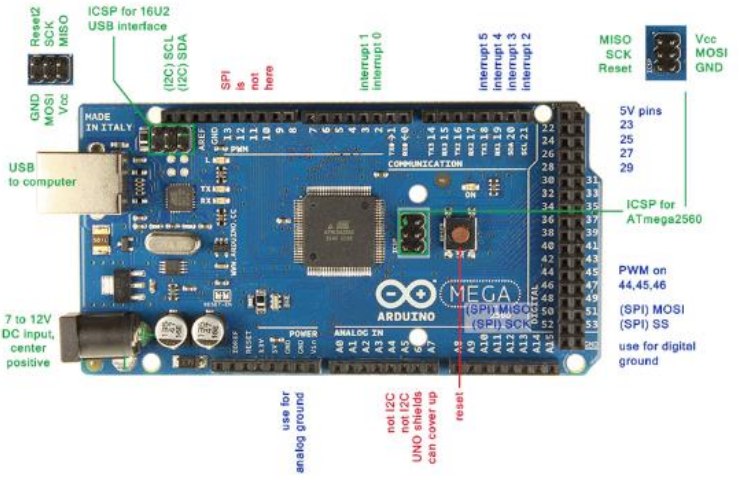
1. Vcc : Pin supply +5 Volt.
2. GND : Pin *ground.*
3. SDA : Pin masukan data SDA.
4. SCL : Pin masukan *clock* SCL.
5. DRDY: *Not Connected*.

Gambar 2.11 Pin kompas digital HMC5883L [12]

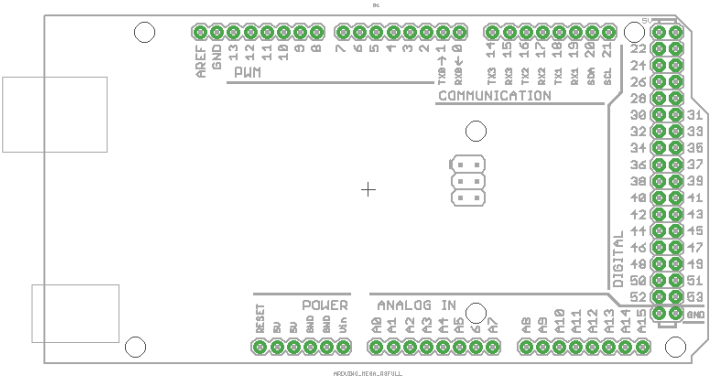
2.4. Mikrokontroller Arduino Mega 2560

Mikrokontroler berfungsi untuk memproses dan mengolah data masukan dan untuk menghasilkan sebuah hasil keluaran yang kita kehendaki. Mikrokontroler Arduino Mega 2560 ditunjukkan pada Gambar 2.12 adalah papan mikrokontroler berbasiskan ATmega2560. Arduino Mega 2560 memiliki 54 pin digital *input/output*, dimana 15 pin dapat digunakan sebagai *output* PWM, 16 pin sebagai *input* analog, dan 4 pin sebagai *Universal Asynchronous Receiver Transmitter* (UART), 16 MHz kristal osilator, koneksi USB, *jack power, header* ICSP, dan tombol reset. ATMega 2560 sendiri memiliki 11 *port* dan memiliki jumlah total pin sebanyak 100 pin. Chip tersebut memiliki pin *Serial Data* (SDA) dan *serial Clock* (SCL), 4 komunikasi *serial* dan *pin change interupt* yang sangat memungkinkan jika digunakan untuk keperluan sistem kendali yang banyak menggunakan sensor dan jalur komunikasi *serial*.

Tegangan yang direkomendasikan untuk men-*supply* Arduino ini berkisar antara 7 Volt hingga 12 Volt. Sedangkan tegangan yang diijinkan antara 6 Volt hingga 20 Volt. Secara keseluruhan pin I/O Arduino Mega 2560 ini ditunjukkan pada Gambar 2.13.



Gambar 2.12 Bentuk fisik Arduino Mega 2560 [13]



Gambar 2.13 Pinout Arduino Mega 2560 [14]

Fungsi dari pin-pin Arduino Mega sebagai berikut:

a. **Vin.** Merupakan tegangan masukan Arduino ketika Arduino menggunakan sumber daya dari luar (sebagai ganti dari sumber daya dari USB atau sumber daya dari regulator).

b. **5V.** Pin keluaran daya dari regulator 5V. Papan ini dapat disupply dengan daya dari *power-jack* (7-12V), USB(5V), memberikan

suppli dari pin 5V, atau 3.3V secara langsung tanpa melewati regulator. Dapat membuat kerusakan pada papan Arduino.

c. **3V3.** Tegangan 3.3V yang dibangkitkan oleh regulator dalam papan. Arus maksimal dari sumber ini adalah 50 mA.

d. **GND.** Pin ground

e. **IOREF.** Pin pada papan Arduino ini menghasilkan tegangan referensi yang digunakan papan Arduino dalam beroperasi. Dapat mengoperasikan logika tegangan pada pin I/O dengan tegangan 3.3V dan 5V.

f. **Aref.** Tegangan referensi yang digunakan untuk referensi *Analog Digital Converter* (ADC).

g. **Reset.** Disediakan untuk menambahkan tombol reset dari luar.

h. **Serial.** Digunakan untuk menerima (Rx) dan untuk mengirim (Tx) TTL data serial.

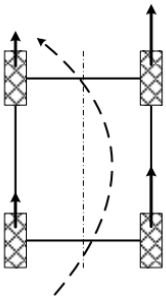
i. **External Interrupt.** Dapat dikonfigurasikan sebagai pin *trigger* sebuah interupsi eksternal, *rising* atau *falling edge*.

j. **PWM.** Menghasilkan keluaran berupan nilai PWM menggunakan fungsi analogWrite().

k. **SPI.** Digunakan untuk komunikasi SPI menggunakan pustaka SPI.

2.5. *Differenstial Speed Steering*

Differential speed steering merupakan metode kendali manuver kendaraan dengan memanfaatkan selisih kecepatan roda kanan dan roda kiri. Mekanisme *steering* dengan metode ini seperti yang ditunjukkan pada Gambar 2.14 membuat kendaraan secara mekanik menjadi sederhana yang mana menghasilkan jalur baru untuk merealisasikan kendali arah untuk kendaraan elektrik [15].



Gambar 2.14 Differential Speed Steering [15]

Untuk mudahnya, dua roda pada kendaraan yang masing-masing diasumsikan berotasi dengan kecepatan yang sama .Untuk *differential speed steering*, arah belokan ditentukan oleh sisi yang mempunyai kecepatan lebih tinggi dari lainnya. Dapat didefinisikan dengan kendaraan akan berbelok ke kiri jika $\Delta\omega > 0$ dan akan berbelok ke kanan jika $\Delta\omega < 0$. Untuk mendapatkan selisih perbedaan kecepatan, strategi kendali dapat menggunakan tiga cara. Yang pertama adalah menambah kecepatan roda terluar, yang kedua mengurangi kecepatan roda terdalam, dan yang ketiga adalah menambah kecepatan pada roda terluar dan mengurangi kecepatan roda terdalam. Persamaan kecepatan sudut yang dihasilkan dari perbedaan kecepatan roda ditunjukkan pada persamaan 2.10 dan 2.11.

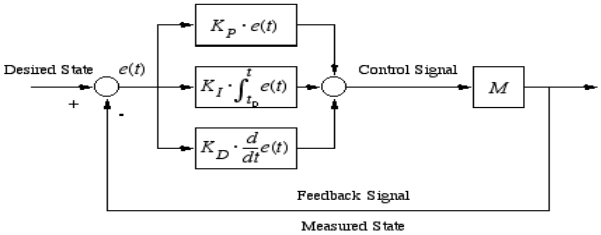
$$\omega_r(t) = \frac{V_r(t)}{r} \tag{2.10}$$

$$\omega_l(t) = \frac{V_l(t)}{r} \tag{2.11}$$

2.6. Kendali Proposional Integral Derivatif

Kendali proporsional integral derivatif adalah suatu sistem kendali yang merupakan gabungan dari kendali proporsional, kendali integral dan kendali derivatif dimana kendali proporsinal akan menghasilkan keluaran kendali yang sebanding dengan nilai eror sehingga diperoleh nilai *steady state* [16]. Sedangkan kendali integral berfungsi untuk meminimalisir nilai *error steady state* terhadap *setpoint*.

Untuk mengurangi osilasi atau *overshoot* respon maka ditambahkan kendali derivatif yang merupakan fungsi derivatif dari nilai *error* dikalikan dengan konstanta derivatif sehingga kendali PID ditunjukkan pada persamaan 2.12, dan diagram bloknya ditunjukkan pada Gambar 2.15 .



Gambar 2.15 Kendali proporsional integral derivatif [16]

$$\mathrm{u}(t) = \mathrm{P}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \quad (2.12)$$

Dari setiap kendali yang menyusun kendali PID mempunyai karakteristik masing-masing.

2.6.1. Kendali Proposional

Kedali proposional adalah kendali yang menghasilkan keluaran kendali yang sebanding dengan *error* masukan yang dikalikan dengan konstanta proposional. Kendali ini berfungsi untuk memperkuat sinyal *error*, sehingga mempercepat keluaran sistem mencapai titik referensi. Persamaan dari kendali proposional ini ditunjukkan pada persamaan 2.13

$$u(t) = K_p\, e(t) \quad (2.13)$$

2.6.2. Kendali Integral

Kendali integral adalah kendali yang menghasilkan keluaran berbanding lurus dengan besar dan lamanya *error*. Integral dalam kendali PID adalah jumlah *error* tiap waktu dan mengakumulasi *offset* yang sebelumnya telah dikoreksi. *Error* terakumulasi dikalikan dengan *gain* integral (Ki) dan menjadi keluaran kendali. Persamaan kendali integral ini ditunjukkan pada persamaan 2.14.

$$u(t) = K_i \int_0^t e(t)\, dt \quad (2.14)$$

Kendali integral mempercepat perpindahan proses menuju *setpoint* dengan menghilangkan *error stady state* yang muncul pada kendali proposional. Namun, karena integral mengakumulasi error sebelumnya, maka dapat menyebabkan *overshoot*.

2.6.3. Kendali Derivatif

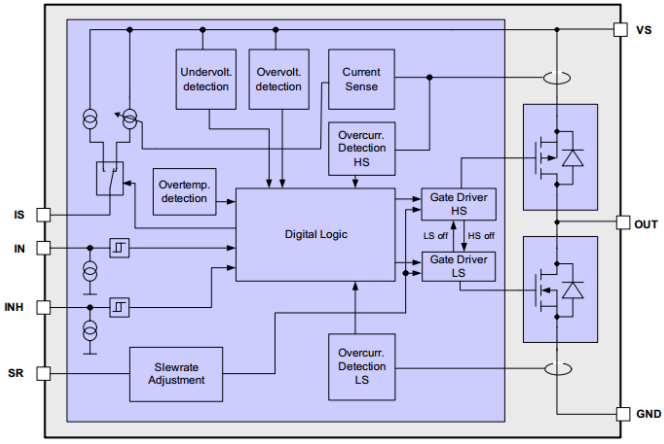
Kendali derifatif adalah kendali yang menghasilkan keluaran sebanding dengan nilai selisih dari masukan. Kendali derivatif digabungkan dengan kendali proposional, akan meredam nilai *overshoot* dari keluaran kendali. Persamaan kendali derivatif ditunjukkan pada persamaan 2.15.

$$u(t) = K_d \tau_d \left(\frac{d_e(t)}{dt}\right) \quad (2.15)$$

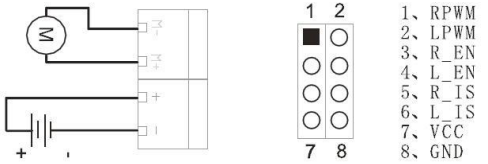
2.7. *Driver* Motor BTN7960

BTN7960 adalah ditunjukkan pada Gambar 2.16 adalah *integrated hight current half bridge* untuk aplikasi *driver* motor. *Driver* ini berisi satu *p-channel* MOSFET pada *highside* dan satu *n-channel* MOSFET pada *lowside* yang terintegrasi menjadi satu. Antarmuka dengan mikrokontroler dibuat mudah oleh produsen dengan fitur *logic level input*, sensor arus, *slew rate adjusmenet*, dan penghasil waktu mati dan proteksi pada kelebihan temperatur, arus, tegangan, hubung pendek dan kurang tegangan. Blok diagram dari *driver* motor ini dapat dilihat pada Gambar 2.17. Tegangan *supply* dari -0,3 Volt hingga 45 Volt. Tegangan *logic* dari -0,3 Volt hingga 5,5 Volt, dengan arus beban maksimal hingga 44 Amper.

Gambar 2.16 Driver motor BTN7960 [17]



Gambar 2.17 Diagram blok driver motor BTN7960 [17]



Gambar 2. 18 Pin masukan logika modul BTN7960 [18]

Modul *driver* motor ini mempunyai pin logika yang dihubungkan dengan kendali, dan pin daya yang dihubungkan dengan motor dan sumber daya untuk motor. Pin logika dan pin daya dari modul ini ditunjukkan pada Gambar 2.18.

1 RPWM : Sinyal masukan PWM R.
2 LPWM : Sinyal masukan PWM L.
3 R_EN : Pin aktivasi sinyal R. 1 = aktif, 0 = tidak aktif.
4 L_EN : Pin aktivasi sinyal L. 1 = aktif, 0 = tidak aktif.
5 R_IS : Pin sensor arus R.
6 L_IS : Pin sensor arus L.
7 VCC : Pin VCC +5V.
8 GND : Pin GND.

M+ : Pin + motor.
M- : Pin – motor.
+ : Sumber daya + motor
- : Sumber daya – motor.

Spesifikasi dari modul *driver* motor BTN7960 ini ditunjukkan pada Tabel 2.6.

Tabel 2. 6 Spesifikasi modul driver motor BTN7960

Parameter	Min	Typ	Max	Unit
Tegangan Supply	-0.3		45	V
Tegangan Masukan Logika	-0.3		5.3	V
Continous Drain Current	-44		44	A
Pulsed Drain Current	-90		90	A
PWM Current	-55		55	A
Arus Supply		2	3	mA

Gambar 2.19 3DR Telemetri [19]

2.8. 3DR Telemetri

3DR yang ditunjukkan pada Gambar 2.19 merupakan salah satu merek alat komunikasi data nirkabel yang menggunakan radio frekuensi dengan frekuensi 433 MHz dan 915 MHz. Modul ini menggunakan protokol *serial* UART. Dalam satu paket, modul ini memiliki sisi *transmitter* dan *receiver*.

Modul ini terbagi menjadi dua bagian, yaitu bagian *ground* dan *air*. Pada bagian *air*, terdapat empat pin, yaitu Rx, Tx, Vcc, dan *ground*. Sedangkan pada sisi *ground*, VCC, Data +, Data -, dan GND.

Sisi *ground* modul ini menggunakan protokol *Universal Serial Bus* (USB) dalam berkomunikasi dengan perangkat (misalnya komputer). Daya maksimum dari modul ini 100 mW, sensitifitas *receiver* -117 dBm, dan mampu digunakan untuk komunikasi *full-duplex* dua arah. Kecepatan transmisi yang didukung oleh modul ini mulai dari 4800 bps hingga 115200 bps. Tegangan *supply* mulai dari 3.7 VDC hingga 6 VDC, arus *transmitting* 100 mA pada 30 dBm, arus penerima 25 mA, dengan level logika serial 3.3 Volt.

Gambar 2. 20 Pin 3DR telemetri [20]

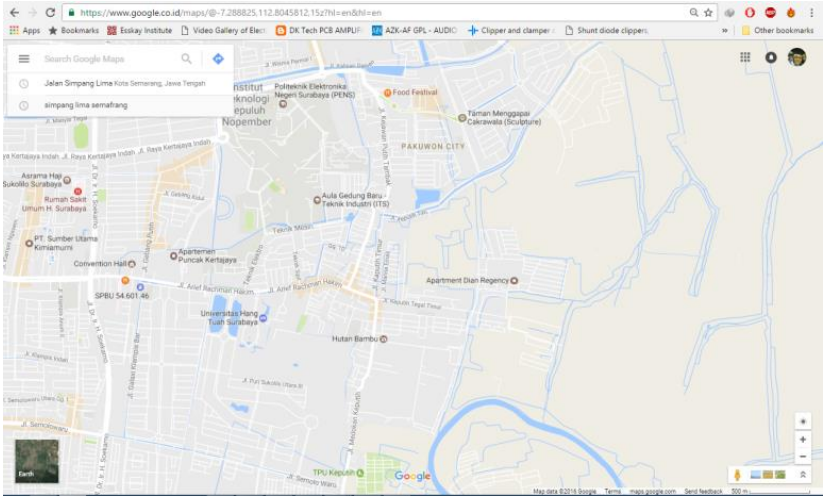
Modul ini mempunyai lima hingga enam pin I/O, tetapi secara umum hanya digunakan empat pin saja. Yaitu pin Rx, Tx, VCC, dan GND. Konfigurasi pin dari modul ini ditunjukkan pada Gambar 2.20. Di mana Rx digunakan untuk menerima data masukan serial, Tx untuk pengiriman data serial, VCC dan GND merupakan pin sumber daya.

2.9. Pemetaan

Istilah pemetaan seringkali digunakan pada ilmu matematika untuk menunjukkan proses pemindahan informasi dari satu bentuk ke bentuk lainnya. Proses tersebut serupa dengan apa yang dilakukan oleh kartografer, yaitu memindahkan informasi dari permukaan bumi ke dalam kertas [21]. Hasil dari pemindahan informasi tersebut dinamakan peta.

2.9.1. Pemetaan Berbasis Google Map

Google Map merupakan fasilitas peta yang dimiliki oleh Google. Layanan ini disediakan oleh Google secara gratis untuk diakses melalui *website maps.google.com*. Tampilan dari Google Map ditunjukkan pada Gambar 2.21. Google Map menyediakan tampilan satelit, *map*, *terrain*, *traffic*, dan *street* dengan kemampuan skala hingga 1:50 meter [22].



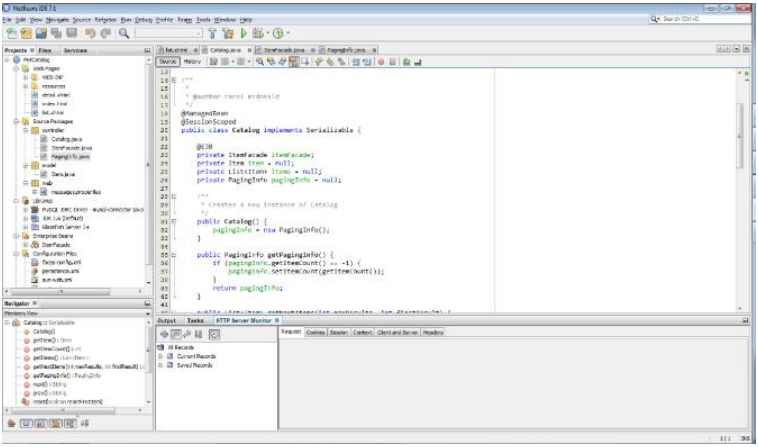
Gambar 2.21 Tampilan Google Map pada Web Browser [22]

Google Map dibuat dengan menggunakan kombinasi dari gambar peta, database, serta obyek-obyek interaktif yang dibuat dengan bahasa pemrograman berbasis web. Gambar-gambar peta yang muncul pada layar merupakan hasil komunikasi dari pengguna dengan *database* pada *webserver* Google.

Google Map juga memberikan kemudahan jika ingin membuat suatu aplikasi yang memanfaatkan peta dari Google ini. Google melengkapi sistem database petanya dengan *Application Programming Interface* (API) sebagai media integrasi aplikasi, antara pengguna (developer) dengan *database* peta milik Google.

2.10. NetBeans *Integrated Development Environtment* (IDE)

NetBeans IDE merupakan *software development platform* yang ditulis dengan bahasa pemrograman *JAVA*. *Platform* NetBeans mengijinkan aplikasi dikembangkan dari satu set komponen software modular yang disebut sebagai modul. Aplikasi berbasis NetBeans, dapat diberi tambahan atau dikembangkan oleh pengembang pihak ketiga.Semula, aplikasi NetBeans IDE ini diperuntukkan demi suatu pengembangan dalam JAVA. Namun, aplikasi ini juga mendukung program-program pembuatan bahasa lain secara khusus seperti PHP, C/C++ dan HTML5 [23].



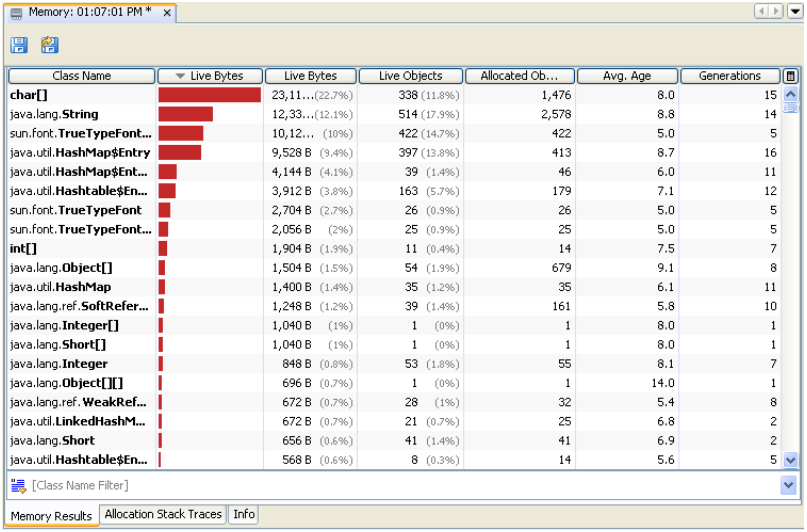
Gambar 2.22 Jendela NetBeans IDE [23]

Platform NetBeans merupakan bingkai kerja untuk menyederhanakan pengembangan aplikasi *desktop* dari *Java Swing*. Jendela NetBeans IDE ditunjukkan pada Gambar 2.22.

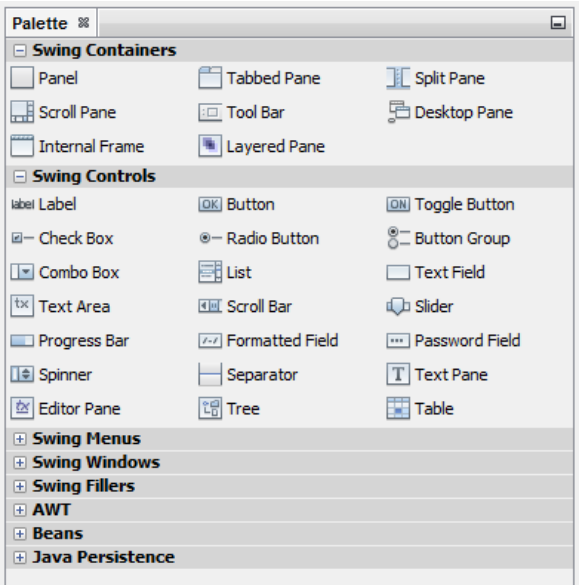
NetBeans IDE mempunyai berbagai kemudahan dalam mengembangkan perangkat lunak. Aplikasi dapat memasang modul secara dinamis. Setiap aplikasi mendapatkan pemberitahuan ada pembaharuan aplikasi atau modul, pengguna dapat mengunduh dan *upgrade* aplikasi secara langsung walaupun ketika aplikasi sedang berjalan. Aplikasi ini juga terbagi menjadi beberapa modul yang terintegrasi.

a. NetBeans Prolifier

NetBeans Prolifier Gambar 2.23 merupakan alat yang digunakan untuk memantau aplikasi Java. Dia membantu pengembang untuk menemukan masalah *memory leaks* dan mengoptimalkan kecepatan program.



Gambar 2.23 NetBeans Prolifier [23]



Gambar 2.24 Peralatan pembuatan GUI [23]

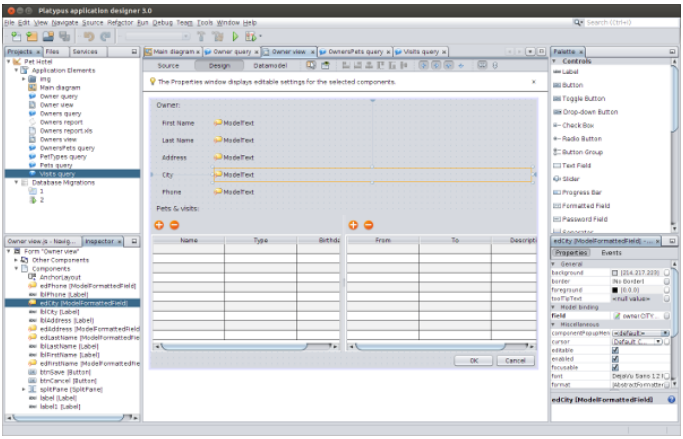
b. *Graphic User Interface* (GUI) Design Tool

Secara formal disebut juga sebagai *Project Matisse*, desain peralatan GUI pada Gambar 2.24 memungkinkan pengembang untuk mendesain antarmuka menggunakan fasilitas *drag* dan *drop.*

c. NetBeans JavaScript Editor

Editor JacaScript NetBeans memberikan dukungan tambahan untuk JavaScript, Ajax, dan CSS. Fitur editor JavaScript terdiri dari sintaks, *refactoring*, penyelesaian kode untuk objek dan fungsi *native*, membangun kelas JavaScript *class*, membangun Ajax callback dari template, dan cek kompatibilitas *browser* otomatis.

Fitur editor CSS terdiri dari penyelesaian kode untuk nama gaya, bernavigasi cepat melalui panel navigator, menampilkan deklarasi aturan CSS dalam daftar *View* dan struktur file di *Tree View*, menyortir tampilan dengan nama, jenis, atau urutan deklarasi (*List and Tree*), menciptakan deklarasi aturan(*Tree only*) dan *refactoring* bagian dari nama aturan(*Tree only*). Tampilan NetBeans JavaScript *editor*ini ditunjukkan pada Gambar 2.25.



Gambar 2.25 NetBeans JavaScript Editor [23]

Halaman Ini Sengaja Dikosongkan

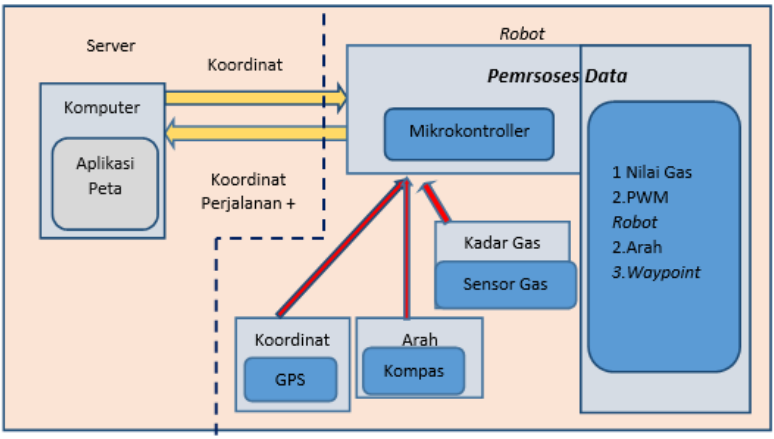
BAB III
PERANCANGAN SISTEM

Pada bab ini menjelaskan tentang perancangan sistem yang meliputi arsitektur sistem, perancangan perangkat keras, perancangan perangkat lunak, desain kendali dan persiapan robot.

3.1 Arsitektur Sistem

Blok diagram dari Gambar 3.1 merupakan penyederhanaan dari sistem pemetaan kadar gas berbahaya yang memanfaatkan *Autonomous Navigation Robot* menggunakan GPS sebagai Tugas Akhir ini. Perencanaan sistem ini bertujuan agar robot dapat bekerja sesuai dengan perencanaan. Yaitu dapat berjalan sesuai arah dari koordinat yang diberikan secara otomatis, mendeteksi nilai gas, dan mengirimkan data nilai gas yang terdeteksi dan koordinat perjalanan yang dilaluinya untuk dikirim ke komputer *server* agar data yang diterima, dapat langsung ditampilkan pada aplikasi peta.

Aplikasi peta mempunyai antarmuka yang mana digunakan untuk memberikan titik-titik *waypoint* yang harus dituju oleh robot, aplikasi peta juga harus menampilkan data yang diterima oleh komputer berupa data koordinat perjalanan robot dan nilai gas berbahaya.

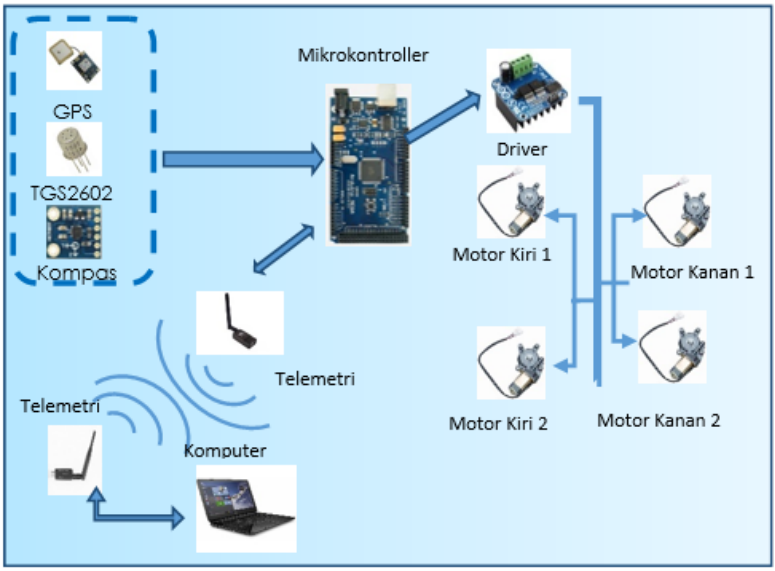


Gambar 3. 1 Diagram blok arsitektur sistem

Pada bagian robot, terdapat beberapa bagian seperti sensor, pemroses data, dan kendali. Pada bagian sensor, terdapat tiga bagian komponen yaitu GPS untuk data koordinat, kompas untuk data arah, dan sensor gas untuk mengetahui kadar gas. Pada bagian pemroses data terdapat mikrokontroler untuk mengolah data masukan dari sensor dan komputer, yang nantinya akan digunakan untuk mengendalikan bagian kendali. Di mana bagian kendali ini terdapat aktuator yang berupa motor DC.

Secara keseluruhan sistem bekerja dengan memasukkan nilai *latitude* dan *longitude* pada aplikasi peta sebagai *waypoint*. Sebelum memasukkan koordinat untuk *waypoint*, robot dan komputer harus terhubung. Untuk menghubungkan antara robot dan komputer, menggunakan modul telemetri nirkabel dengan komunikasi serial. Kemudian menekan tombol 'menambah' untuk menambah *waypoint* dan menekan tombol 'masukkan' untuk memasukkan data *waypoint*. Data *waypoint* ini akan dikirim dari komputer ke dalam robot melalui komunikasi serial. Di sisi lain, aplikasi peta juga akan menampilkan titik *waypoint* yang dimasukkan. Setelah beberapa titik *waypoint* dimasukkan ke dalam robot, robot akan berjalan menuju titik *waypoint* ketika tombol 'mulai' ditekan.

Ketika robot berjalan, data *waypoint* yang telah masuk, secara urut akan menjadi titik-titik tujuan yang harus dilalui. Sensor GPS secara periodik memberikan informasi data lokasi robot berada, dengan bantuan sensor kompas yang memberikan arah robot. Informasi data tersebut masuk ke dalam pemroses data, selanjutnya data tersebut digunakan sebagai acuan. Koordinat lokasi menjadi data jarak dan arah yang harus dituju antara *waypoint* dan robot, data kompas menjadi arah *heading* robot ketika menuju *waypoint*. Data kompas juga memberikan nilai selisih sudut yang digunakan kendali untuk mengatur PWM motor, agar robot dapat bermanuver kearah sudut yang harus dituju. Data lokasi juga digunakan kendali untuk memutuskan robot harus menuju ke *waypoint* ke berapa. Skematik dari sistem yang ditunjukkan pada Gambar 3.2 ada di lampiran.

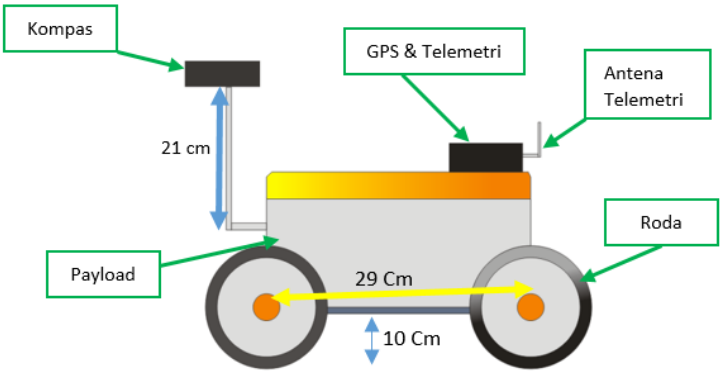


Gambar 3. 2 Perancangan sistem perangkat keras

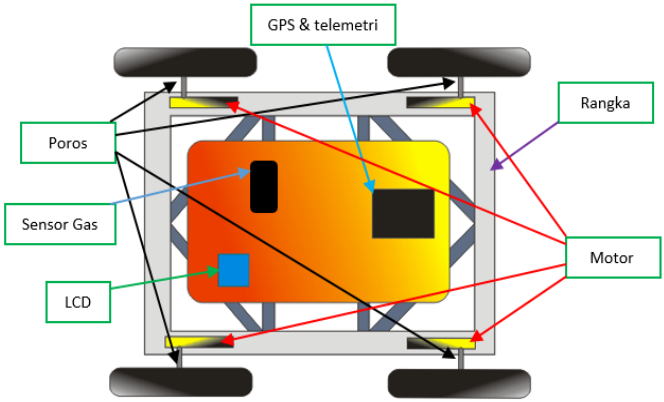
3.2 Desain Mekanik Robot

Desain mekanik robot ini dimulai dengan perancangan perangkat keras sistem kendali *Autonomous Navigation Robot* penggabungan fungsi beberapa komponen seperti ditunjukkan pada Gambar 3.2 yaitu, telemetri, *driver motor*, sensor-sensor, mikrokontroler, dan aktuator. Semua komponen penyusun dirangkai sedemikian rupa seperti yang ditunjukkan pada Gambar 3.3 dan Gambar 3.5 sebagai desain robot keseluruhan.

Pembuatan mekanik robot menggunakan desain yang dibuat oleh penulis. Model robot yang digunakan adalah mobil robot berpenggerak motor DC 4 buah dengan masing-masing menggunakan satu roda pada tiap motor



Gambar 3. 3 Desain mekanik tampak samping

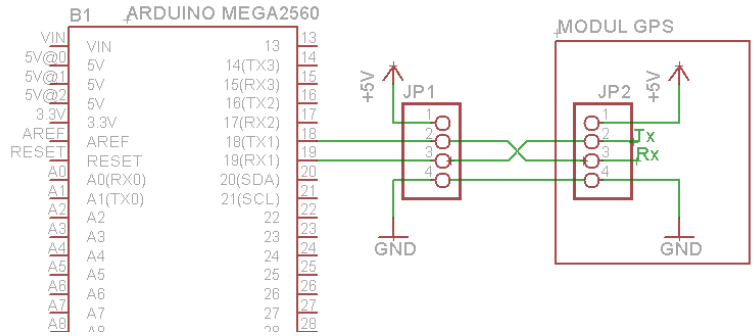


Gambar 3. 4 Desain mekanik tampak atas

3.2.1. Perancangan Sensor GPS Ublox M8

GPS Ublox M8 menggunakan komunikasi serial UART TTL yang dihubungkan pada pin serial1 arduino. Dengan model *cross connection* di mana Rx GPS terhubung Tx arduino, dan Tx GPS terhubung Rx arduino.

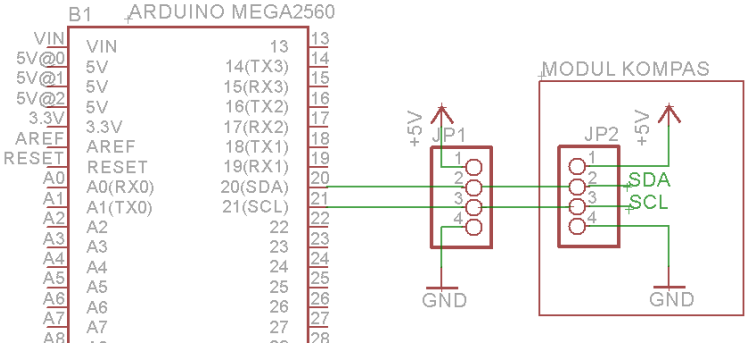
Skema jalur konfigurasi rangkaian GPS Ublox M8 dengan Arduino Mega dapat dilihat pada Gambar 3.5.



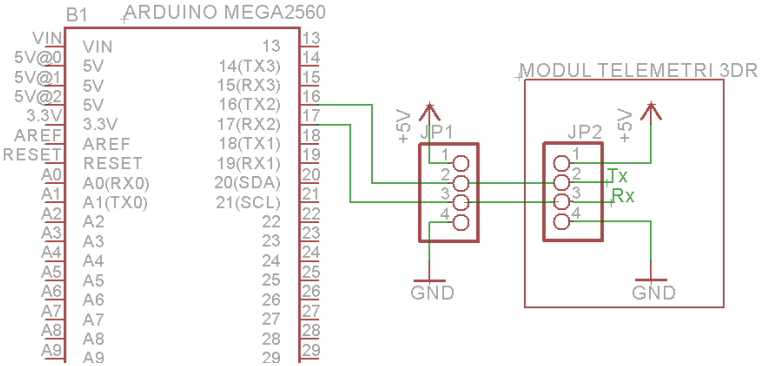
Gambar 3. 5 Koneksi modul GPS dengan Arduino Mega 2560

3.2.2. Perancangan Sensor Kompas HMC5883L

Sensor kompas digunakan sebagai arah *heading* robot. Sensor ini menggunakan komunikasi *Inter-Integrated Communication* (I2C). Keunggunlan dari komunikasi I2C ini adalah dapat menampung banyak perangkat hanya dengan menggunakan dua buah kabel komunikasi. Kelemahannya adalah kecepatan transfer data yang cukup lambat, mulai dari 100 kHz hingga 400 kHz. Konfigurasi rangkaian sensor kompas HMC5883L dengan Arduino Mega ditunjukkan pada Gambar 3.6.



Gambar 3. 6 Konfigurasi modul sensor kompas HMC5883L



Gambar 3. 7 Konfigurasi modul telemetri 3DR

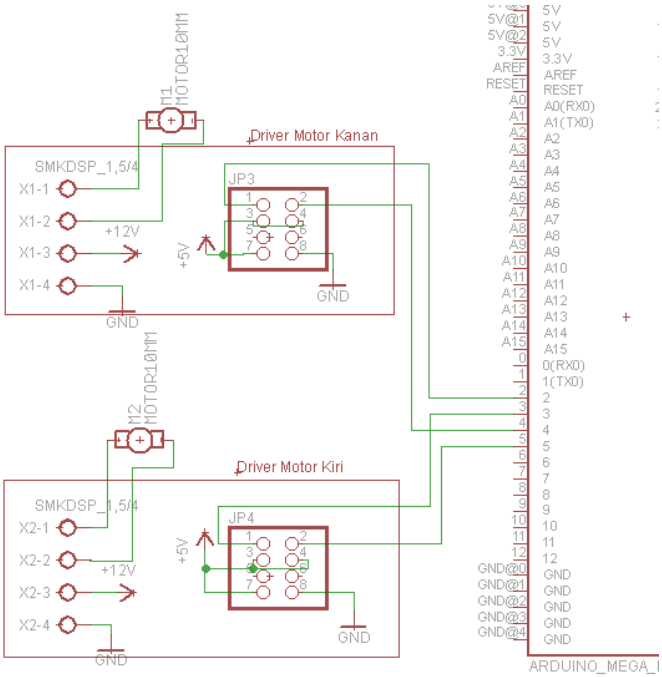
3.2.3. Perancangan Rangkaian Transceiver 3DR

Transceiver 3DR merupakan modul telemetri dengan fasilitas komunikasi *two waysfull duplex.*

Dengan basis Serial TTL UART pada sisi *air* dan Serial TTL UART to USB pada sisi *ground* yang menggunakan IC CP2102 sebagai konverternya. Secara *default* modul ini menggunakan *through connection* yang mana Rx modul terhubung dengan Rx arduino, dan Tx modul terhubung dengan Tx Arduino. Pada rancangan ini, modul ini dihubungkan pada serial2 Arduino. Konfigurasi rangkaian telemetri ini dengan Arduino ditunjukkan pada Gambar 3.7.

3.2.4. Perancangan Rangkaian *Driver* Motor BTN7960

Modul driver motor BTN7960 merupakan modul *H-Bridge* yang berbasis dua IC BTN7960 Half-Bridge yang dikombinasikan menjadi satu. Konfigurasi dari rangkaian *driver* motor dengan arduino ditunjukkan pada Gambar 3.8.

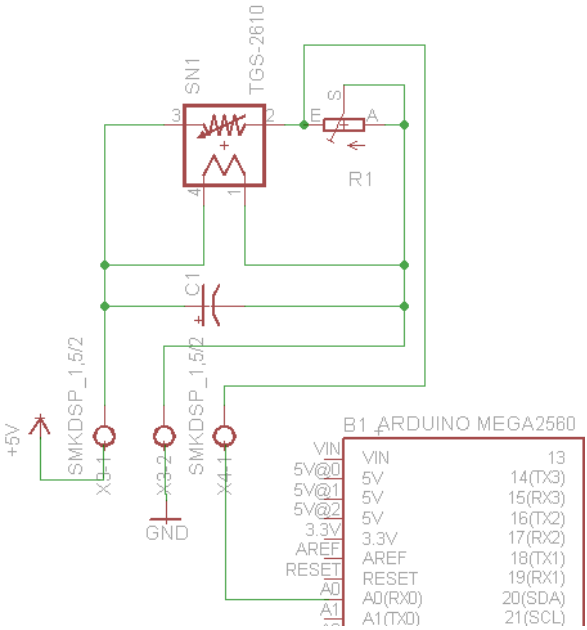


Gambar 3. 8 Konfigurasi rangkaian driver motor

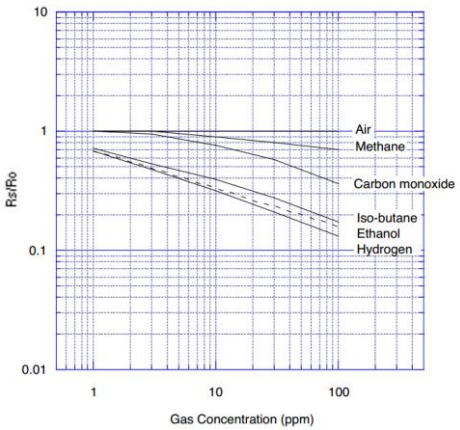
3.2.5. Perancangan Sensor Gas TGS2600

Sensor gas TGS2600 mempunyai hasil keluaran berupa tegangan jika mendeteksi kadar gas polutan di dalam udara. Pin keluaran dari sensor gas ini dihubungkan dengan pin A0 pada Arduino. Sensor ini disupply dengan tegangan 5 volt. Vh yang merupakan sumber *supply* untuk resistor pemanas, dihubungkan paralel dengan Vc. Pemanas ini digunakan untuk memperbaiki elemen sensor pada temperatur tertentu untuk penginderaan secara optimal.

Pada gambar 3.9, pin 1 dan pin 4 yang merupakan sumber daya untuk Resistor pemanas atau Rh, dihubungkan paralel dengan sumber daya Vc. Ini diperlukan karena rangkaian pemanas digunakan untuk memperbaiki elemen pengindera pada temperatur tertentu yang mana mengoptimalkan penginderaan.



Gambar 3. 9 Skema rangkaian sensor gas TGS2600 dengan Arduino



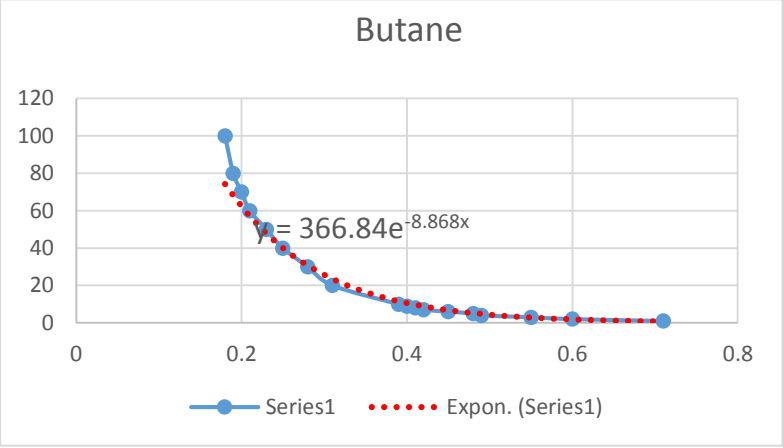
Gambar 3. 10 Grafik kadar ppm iso-butane

Untuk mendapatkan nilai kadar gas, merujuka pada grafik sensor gas yang digunakan. Dalam penelitian ini menggunakan data grafik sensor TGS2600 pada nilai penginderaan gas *iso-butane*. Nilai dari grafik pada Gambar 3.10 kemudian diambil dan dimasukkan pada tabel. Sehingga mendapatkan tabel perbandingan antara Ro/Rs dengan ppm gas seperti yang ditunjukkan pada Tabel 3.1.

Tabel 3. 1 Nilai ppm gas terhadap perbandingan Ro/Rs

Ro/Rs	ppm
0.71	1
0.6	2
0.55	3
0.49	4
0.48	5
0.45	6
0.42	7
0.41	8
0.4	9

Ro/Rs	ppm
0.39	10
0.31	20
0.28	30
0.25	40
0.23	50
0.21	60
0.2	70
0.19	80
0.18	100



Gambar 3. 11 Grafik persamaan kadar ppm gas

Nilai dari Tabel 3.1 kemudian di-*plot* pada grafik, untuk menghasilkan nilai persamaan nilai kadar gas. Melalui persamaan yang dihasilkan oleh grafik pada Gambar 3.11.

3.3 Perancangan Perangkat Lunak

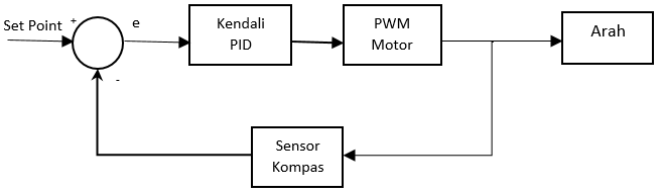
Perancangan perangkat lunak ini digunakan untuk mengintegrasikan dan memfungsikan modul-modul perangkat keras dengan pengguna. Perancangan ini meliputi penunjukkan arah dari GPS, mendapatkan garis lintang dan bujur dari GPS, sudut *waypoint* yang dituju, dan aplikasi peta.

3.3.1. Desain Kendali PID pada *Differential Speed Steering*

Kendali ini digunakan robot dalam bermanuver mengikuti sudut acuan dari hasil pengolahan data koordinat yang diberikan oleh *user* pada robot. Diagram blok dari kendali ini ditunjukkan pada Gambar 3.12. Nilai *setpoint* didapatkan dari fungsi $\tan^{-1}(y/x)$ selisih dua titik antara koordinat lokasi robot dengan koordinat *waypoint*. Dimana y = bujur tujuan – bujur sekarang, dan x = lintang tujuan – lintang sekarang.

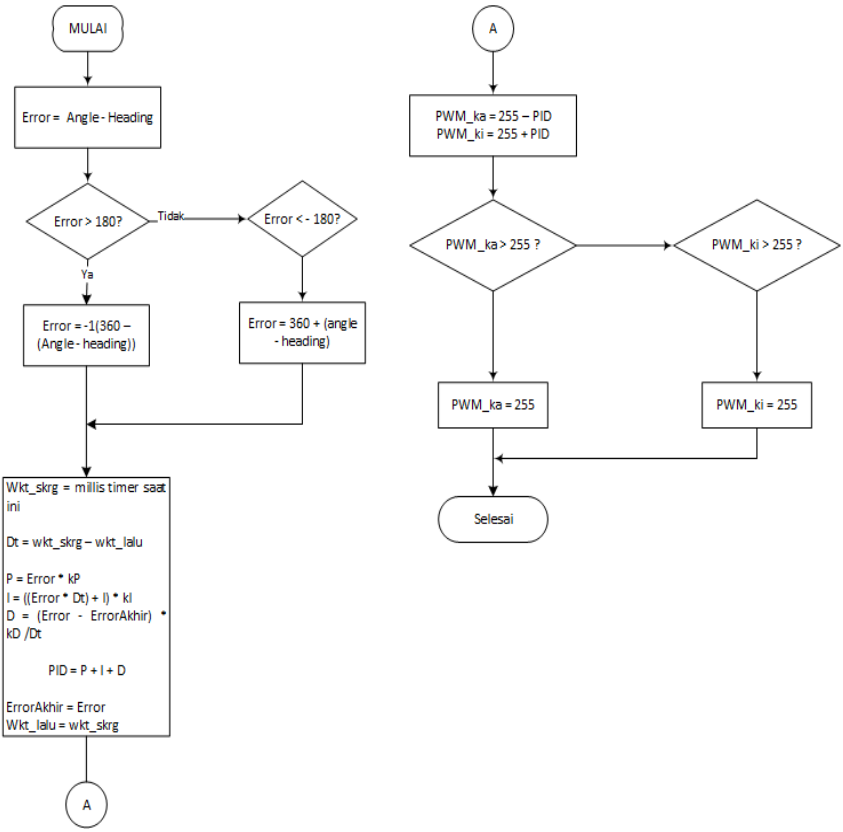
Fungsi tersebut akan menghasilkan nilai antara -3,14/2 hingga 3,14/2. Nilai tersebut kemudian dikonversi ke sudut dengan cara mengalikannya dengan $180/\pi$. Kode dari penjelasan diagram blok adalah sebagai berikut:

```
float hitungSudut = atan2((lon[tujuan] - lon_skrg),
(lat[tujuan] - lat_skrg));//sudul alfa, arah angle set point
  float rubahKeRadian = 57.29577951; //ubah radian ke sudut
  hitungSudut *= rubahKeRadian;
     if (hitungSudut < 0){
      hitungSudut = 360 + hitungSudut;
      }
     else if (hitungSudut > 0){
      hitungSudut = hitungSudut;
      }
     else if (hitungSudut > 360){
      hitungSudut = 0;
      }

     Angle = hitungSudut;
```



Gambar 3. 12 Diagram blok kendali arah motor

Setelah nilai *angle* sebagai nilai *setpoint* didapatkan, kemudian menyusun algoritma agar robot dapat bermanuver ke arah sudut *angle* tersebut. Algoritma itu disusun menjadi diagram alir yang ditunjukkan pada Gambar 3.13.



Gambar 3. 13 Diagram alir algoritma kendali manuver arah

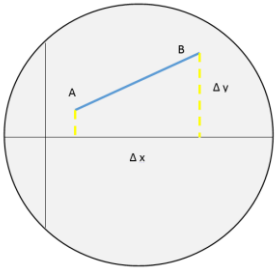
Algoritma yang telah disusun menjadi diagram blok, kemudian diterjemahkan ke dalam kode program. Kode program dari algoritma pada Gambar 3.13 adalah sebagai berikut:

```
Error = Angle – heading;

  if (Error > 180) {
      Error = -1 * (360.00 – (Angle – heading));
  }
  else if (Error < -180) {
      Error = (360.00 + (Angle – heading));
  }

  waktuSekarang = millis();
```

```
  Dt = (float)(waktuSekarang - waktuLalu); //Delta waktu,
waktuSkrg - waktuYgLalu

  float P = Error * kP;
  float I = ((Error*Dt) + I) * kI;
  float D = ((Error - lastError) * kD)/Dt;

  PID = P + I + D;
  lastError = Error;
  //Serial.println("Error=" + String(Error) + " " +
"LastError=" + String(lastError) + " " + "PID=" +
String(jambu));
  waktuLalu = waktuSekarang; //jadi waktuYgLalu

   nilaiPWM_ka = (255 - PID);
   nilaiPWM_ki = (255 + PID);

  if(nilaiPWM_ka > PWM_MAX){
      nilaiPWM_ka = PWM_MAX;
        if (nilaiPWM_ki < 0){
            kiriTajam();
          }
       else {
          maju();
        }
    }

  if(nilaiPWM_ki > PWM_MAX){
      nilaiPWM_ki = PWM_MAX;
        if(nilaiPWM_ka < 0){
            kananTajam();
          }
         else{
            maju();
        }
    }
  #ifdef _DEBUG_
  Serial.println("PID = " + String(millis()));
  #endif
```

3.3.2. Navigasi ke *Waypoint*

Titik *waypoint* didapatkan dari masukan dari pengguna dengan memberikan nilai-nilai *latitude* dan *longitude*.

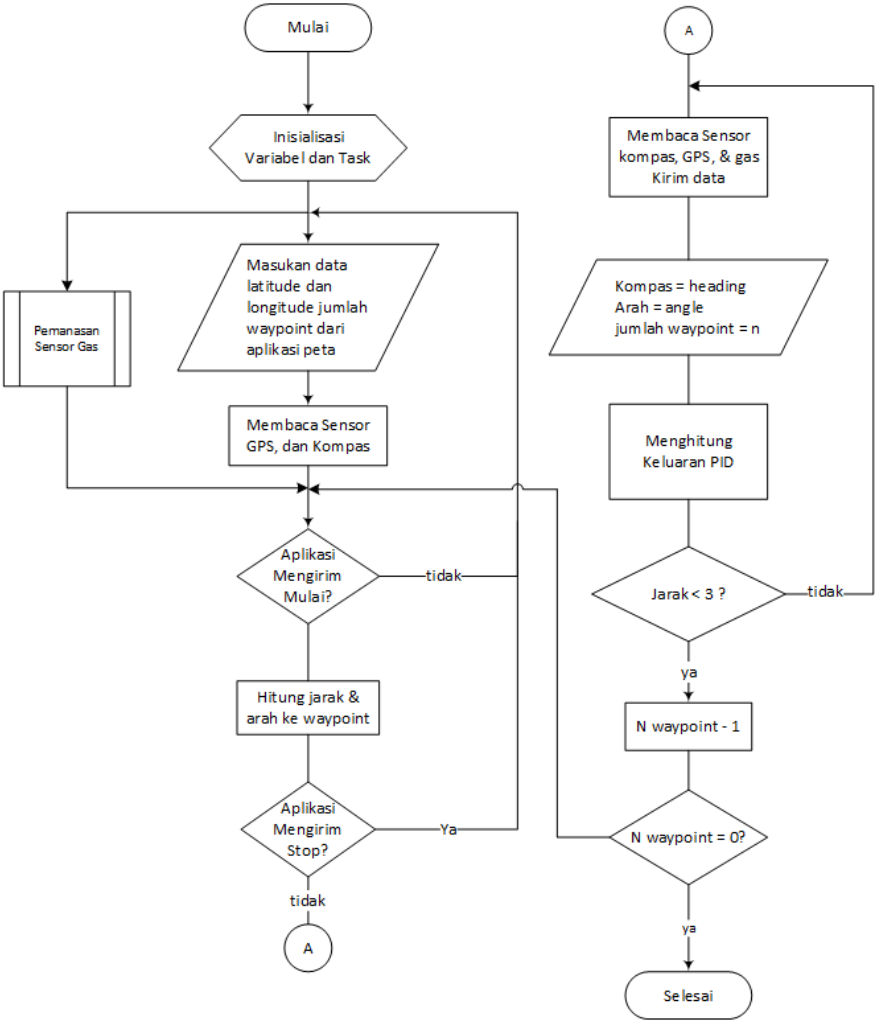


Gambar 3. 14 Ilustrasi jarak titik A ke B

Dengan menggunakan *formula phytagoras*, akan didapatkan jarak antara 2 titik. Di mana titik A dapat direpresentasikan sebagai titik asal, dan titik B sebagai titik tujuan. Jarak titik A ke B dapat dicari menggunakan formula $\sqrt{\Delta x^2 + \Delta y^2}$ di mana Δx adalah selisih antara lintang sekarang dikurangi lintang tujuan, dan Δy adalah selisih antara bujur sekarang dikurangi bujur tujuan.

Jika sudah diketahui jarak, maka dapat disusun algoritma *waypoint*. Sebagai ilustrasi, jarak titik A dan B ditunjukkan pada Gambar 3.14. Selanjutnya menyusun algoritma diagram alir yang ditunjukkan Gambar 3.15. Pada diagram alir Gambar 3.15, ketika robot mulai, pertama kali menginisialisasi data masukan berupa koordinat *waypoint* yang dikirim oleh *server*, secara bersamaan memanaskan sensor gas, dan mengaktifkan *task* dari tiap bagian program. Setelah *task* dari tiap tugas aktif, maka program akan mulai membaca nilai sensor GPS, dan kompas. Ketika pembacaan sensor pada awal, program menunggu masukan dari *server*, apakah ada perintah untuk mulai. Jika iya, maka semua *task* pembacaan sensor dan navigasi aktif. *Task* navigasi mengukur jarak dan arah robot terhadap *heading* dan tujuan. Dalam urutan tersebut, terdapat kondisi pemeriksaan untuk menghentikan segala aktifitas robot. Dimana jika *server* mengirimkan data untuk berhenti, maka robot akan berhenti dan semua urutan program akan mulai dari awal. Jika tidak, maka selanjutnya adalah proses membaca sensor GPS, dan gas untuk mengetahui lokasi robot berada dan kadar gas pada lokasi tersebut. Yang kemudian dikirim ke *server*. Setelah itu, data hasil pembacaan sensor GPS dan kompas akan menjadi *setpoint* robot sebagai parameter PID. Hasil dari pemrosesan PID kemudian diperiksa, apakah jarak kurang dari 3 meter terhadap *waypoint*, jika iya, maka akan melanjutkan ke *waypoint* berikutnya. Jika tidak, akan langsung menuju pemeriksaan apakah jumlah *waypoint* = 0. Jika iya maka

robot berhenti. Jika tidak, robot kembali ke *task* menghitung jarak dan arah ke *waypoint*.



Gambar 3. 15 Diagram alir navigasi ke waypoint

Untuk menjalankan algoritma diagram alir dari Gambar 3.15, setiap perangkat modul mempunyai cara akses yang berbeda-beda. Cara akses dari masing-masing modul adalah sebagai berikut:

a. Mengakses Modul GPS Ublox Neo M8

Untuk mendapatkan data GPS, baudrate antara modul dengan penerima harus sama. GPS di sini menggunakan baudrate 9600. Kode dasar untuk membaca data masukan dari jalur Serial1 adalah sebagai berikut:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial1.begin(9600);
}
```

Kemudian data yang telah diterima, perlu dipilah. Karena data yang diterima, masih berbentuk data mentah yang terdiri dari beberapa informasi. Untuk mendapatkan data yang kita perlukan, proses parsing perlu dilakukan. Kode proses parsing memilah data yang berawalan dengan karakter $, dan diakhiri dengan karakter \n adalah sebagai berikut:

```
void loop() {
  char index;
    while (Serial1.available()) {
      index = Serial1.read();
      if(index == '$' || GPSIndex >= 80){
        GPSIndex = 0;
      }
      if(index != '\r'){
        GPSBuffer[GPSIndex++] = index;
      }
      if(index == '\n'){
    ProsesGpsLine();
    GPSIndex = 0;}
    }
```

Semua data yang berawalan karakter $, akan ditampung dalam satu variabel array. Kemudian proses selanjutnya memilah data dengan ID GNGGA dan memilah data koordinat.

```
If((GPSBuffer[1]=='G')&&(GPSBuffer[2]=='N')&&(GPSBuffer[3]==
   'G') && (GPSBuffer[4]=='G') && (GPSBuffer[5]=='A'))

////$GNGGA,055850.00,0717.46098,S,11248.33858,E,1,08,1.56,17
   .4,M,17.3,M,,*66
////   |  0 | 1  |2|  1  |2|3|4 | 5 |  6  |
////$GNGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46
   .9,M,,*47
///                      =====  ← altitude pada urutan 8
for (i=13, j=0, k=0; (I <= GPSIndex) && (j<9); i++){
  if(GPSBuffer[i]==','){
    j++;
  }
  else{
     if(j== 1){
         char inChar = GPSBuffer[i];
      if(GPSBuffer[i] >= '0' && GPSBuffer[i] <= '12'){
            latt += inChar;
            latt.toCharArray(bufflat, 13, 0);
            }
      }
      if(j == 3){
          char inChar2 = GPSBuffer[i];
      if(GPSBuffer[i] >= '0' && GPSBuffer[i] <= '12'){
            lonn += inChar2;
            lonn.toCharArray(bufflon, 13, 0);
            }
        }
    }
  }
  //bufflat = 0712312312; lati = 7.12312312; latti = 7
  lati = atof(bufflat)/10000000;//data char bufflat diubah
   menjadi float. Dibagi utk dpt derajat.
  Int latti = lati; //nilai lati diubah ke integer utk
   hilangkan angka belakang koma
  lat = lati - latti; //lat = 7.12312312 - 7; = 0.12312312
   //dapat data menit
```

```
loni = atof(bufflon)/10000000;
int lonni = loni;
lon = loni – lonni;

//ubah data ke derajat desimal
double latmin = (lat*100)/60; //data menit 0.123; dikali 100
 / 60 → ddddd
latitude = latti + latmin;
double lonmin = (lon*100)/60;
longitude = lonni + lonmin;

Serial.println(latitude,6);
Serial.println(longitude,6);
```

Data latitude dan longitude yang berupa sudut dan menit, kemudian diubah menjadi sudut desimal dengan membagi 10000000. Kemudian diperoleh nilai desimal dengan 7 angka dibelakang koma. Nilai ini kemudian diubah menjadi tipe data Integer untuk menghilangkan nilai dibelakang koma. Kemudian nilai hasil konversi integer digunakan untuk mengurangi nilai float dan menghasilkan nilai menit. Nilai menit dikalikan 100 agar koma mundur, dan dibagi dengan 60, agar mendapat nilai derajat.

b. Mengakses Modul Kompas HMC5883L

Modul kompas HMC5883L menggunakan komunikasi I2C (Inter Integrated Circuit) sebagai protokolnya. Sehingga untuk membaca modul ini, harus mengetahui alamat dari setiap *register* yang akan diakses. Alamat *register* HMC5883L 0x1E, alamat *register* tulis 0x3C, alamat *register* baca 0x3D, alamat *register* data 0x03. Prosedur kode pembacaan sensor kompas HMC5883L adalah sebagai berikut:

```
void kompas(){
  Wire.beginTransmission(0x1E);
  Wire.write(0x03);
  Wire.endTransmission();
  Wire.requestFrom(0x1E, 6);
  if(6 <= Wire.available()){
    x = Wire.read()<<8; //X msb
    x |= Wire.read(); //X lsb
    z = Wire.read()<<8; //Z msb
    z |= Wire.read(); //Z lsb
```

```
    y = Wire.read()<<8; //Y msb
    y |= Wire.read(); //Y lsb
  }

  heading = atan2(y,x);
  float sudutDeklinasi = 0.022;
  heading += sudutDeklinasi;

  //koreksi arah Sudut ketika tanda berubah
  if(heading < 0)  heading += 2*PI;
  //memeriksa arah ketika terselimuti karena penambahan
deklinasi
  if(heading > 2*PI) heading -= 2*PI;
  sudutHeading = heading * 180/M_PI;
```

Data dari masing-masing sumbu mempunyai panjang 16 bit. Pembacaan dibagi menjadi dua, yaitu pembacaan MSB sepanjang 8 bit pertama, kemudian LSB pada 8 bit kedua. Data dari sumbu X dan sumbu Y kemudian dimasukkan dalam formula *atan2(y, x)* untuk menghasilkan nilai *arc tangen* dalam empat kuadran pada sumbu X atau Y tertentu. Sumbu Z tidak digunakan dalam formula, karena dianggap kompas selalu berada dalam keadaan tegak terhadap garis normal. Hasil formula tersebut menghasilkan nilai sudut dengan satuan *radian*dengan nilai antara -3,14/2 hingga 3,14/2. Untuk merubahnya ke bentuk derajat hasil dari formula tersebut dikalikan dengan hasil 180/π.

c. Mengakses Sensor Gas TGS2600

Dari hasil plotting yang menghasilkan persamaan pada Gambar 3.11, maka kode untuk pembacaan gas adalah sebagai berikut:

```
void gasCallback(){
  ADC = analogRead(PIN_ADC);
  float const eksponen = 2.718282;
  float batas = nilaiAwalADC;
  float level = map(ADC, batas, 1023, 0, 100); //kadar
persen

  float tegangan = ADC * (5.0/1023.0);
  float Rs = (4.9 * 10000.0)/tegangan;
  Rs = Rs - 10000.0;

  float perbandingan = Rs/Ro;
  float nilai = (-8.868) * perbandingan;
  float nilaiEksponen = pow(eksponen, nilai);
  kadarGas = 366.84 * nilaiEksponen;
  if(kadarGas < 0){kadarGas = 0;}
```

```
  //Serial.println("Gas = " + String(kadarGas) + ", ADC =
"   +   String(nilaiAwalADC)   +   ",   Ro/Rs   =   "   +
String(perbandingan) + ", Millis = " + String(millis()));

  Serial2.println("$kG,"  +  String(kadarGas)  +  ","  +
String(nilaiAwalADC) + "," + String(perbandingan) + "," +
String(millis()));

  #ifdef _DEBUG_
  Serial.println(String(kadarGas)      +      ","      +
String(nilaiAwalADC) + "," + String(perbandingan) + "," +
String(millis()));
  #endif
  }
```

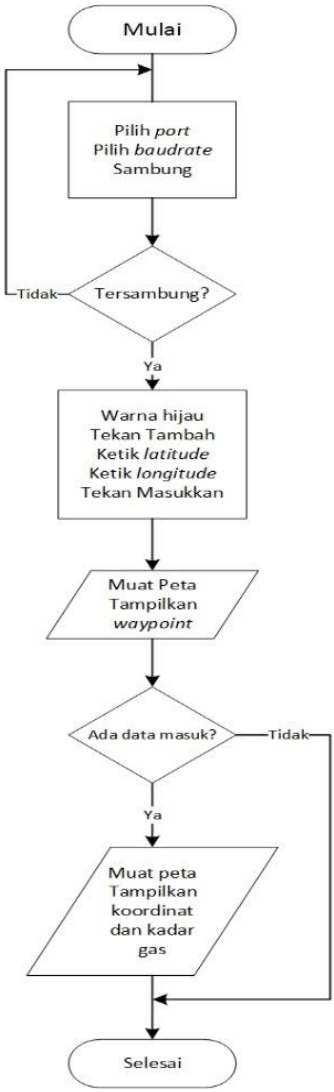
d. Mengakses *Driver* Motor BTN7960

Untuk dapat mengakses perancangan modul *driver* motor BTN7960 pada Arduino seperti yang ditunjukkan pada Gambar 3.9, secara sederhana kode tersebut ditulis sebagai berikut:

```
#define PIN_R_MOTOR_KA  2
#define PIN_L_MOTOR_KA  4
#define PIN_R_MOTOR_KI  3
#define PIN_L_MOTOR_KI  5

void motor(){
  analogWrite(PIN_R_MOTOR_KA, nilaiR1);
  analogWrite(PIN_L_MOTOR_KA, nilaiL1);

  analogWrite(PIN_R_MOTOR_KI, nilaiR2);
  analogWrite(PIN_L_MOTOR_KI, nilaiL2);
}
```

3.3.3. Desain Aplikasi Peta

Desain aplikasi peta di sini merupakan aplikasi yang dapat menampilkan lokasi robot berada dan kadar gas yang terdapat pada lokasi tersebut. Selain itu, aplikasi ini juga dapat digunakan sebagai media untuk memberikan titik-titik *waypoint* pada robot dengan antarmuka yang mudah. Aplikasi peta dijalankan pada komputer *server*.



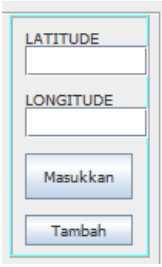
Gambar 3. 16 Diagram alir algoritma aplikasi peta

Untuk dapat berkomunikasi dengan robot, komputer dan robot dihubungkan melalui telemetri dengan protokol serial UART. Agar desain aplikasi dapat bekerja dengan urut, maka pembuatan aplikasi berdasarkan rancangan diagram alir seperti yang ditunjukkan pada Gambar 3.16.

Antarmuka aplikasi peta terdiri dari beberapa panel. Yaitu panel *serial* digunakan untuk memilih *port* dan *baudrate* perangkat telemetri. Panel *waypoint* digunakan untuk memasukkan *latitude* dan *longitude waypoint*. Panel data mengetahui data yang masuk. Panel peta untuk menampilkan peta. Tombol 'tambah' untuk menambah jumlah *waypoint* dan tombol 'masukkan' digunakan untuk mengirim data *latitude* dan *longitude waypoint* ke robot.

a. Panel Serial
   Panel *serial* pada Gambar 3.16 digunakan sebagai jalur komunikasi antara perangkat komputer dengan robot. Secara otomatis panel akan memberikan *list port Com* yang tersedia. Pada pilihan *baudrate* list kecepatan transfer data mulai dari 4800 hingga 115200. Tombol mulai dan *stop* digunakan untuk memberikan perintah mulai berjalan dan berhenti berjalan pada robot.

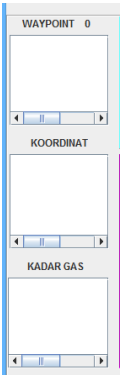
Gambar 3. 17 Panel komunikasi serial



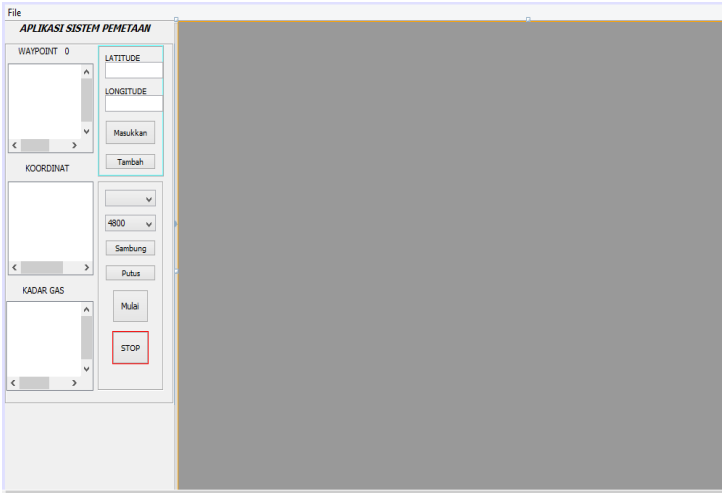
Gambar 3. 18 Tampilan panel waypoint

b. Panel *Waypoint*
Panel *waypoint* pada Gambar 3.17 terdiri dari masukan *latitude* dan *longitude*. Kedua paramater ini dimasukkan secara manual dengan mengetik nilai *latitude* dan *longitude* yang diinginkan. Tombol 'tambah' digunakan untuk menambahkan jumlah *waypoint*. Setelah *waypoint* ditambahkan, kemudian tombol 'masukkan' digunakan untuk menampilkan titik *waypoint* ke peta. Jika koneksi antara aplikasi dan robot terhubung, maka titik tersebut juga akan dikirimkan ke robot.

c. Panel Data
Panel data pada Gambar 3.18 terdiri dari tiga buah *textfield* yang menampilkan data masukan, yaitu data *waypoint*, data koordinat perjalanan, dan data gas. Hanya data dengan awalan karakter $ yang dapat tampil pada panel data ini.



Gambar 3. 19 Panel data



Gambar 3. 20 Rancangan antarmuka aplikasi peta

Secara keseluruhan tampilan desain dari antarmuka aplikasi peta seperti yang ditunjukkan pada Gambar 3.19. Algoritma dari peinsip kerja aplikasi ini ditunjukkan pada Gambar 3.20.

Ketika aplikasi mulai dijalankan, proses pertama kali adalah memilih *port serial* dan *baudrate* dari perangkat. Jika perangkat telah terhubung, maka *background* dari panel komunikasi akan berubah warna menjadi hijau dari yang semula berwarna *magenta*. Langkah selanjutnya adalah memasukkan nilai *latitude* dan *longitude waypoint* melalui panel *waypoint* dan aplikasi akan menampilkan titik koordinat *waypoint* pada panel peta. Ketika ada data masuk ke dalam aplikasi, maka data tersebut akan ditampilkan pada panel peta. Jika tidak, maka proses selesai.

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

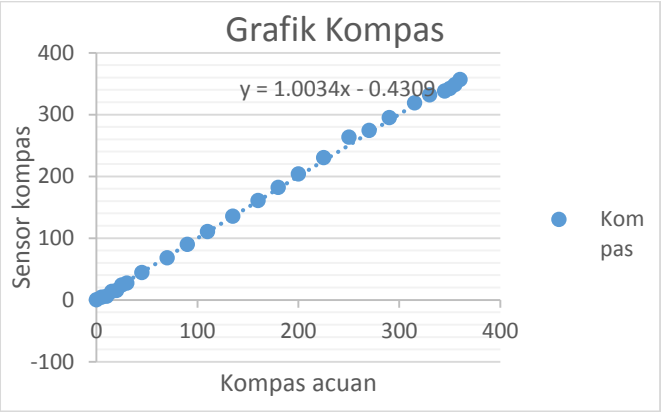
Bab ini menjelaskan tentang pengujian dan analisa sistem dari hasil rancangan yang telah dijelaskan pada bab sebelumnya. Pengujian yang dilakukan adalah pengujian sensor kompas, sensor GPS, sensor gas, PWM kanan, PWM kiri, pengujian program aplikasi dan pengujian navigasi *waypoint*.

4.1 Pengujian Sudut Kompas

Pengujian sensor kompas ini dilakukan dengan cara membandingkan sensor kompas HMC5883L dengan sensor kompas acuan. Sensor kompas HMC5883L dan sensor kompas acuan dipasang dengan sudut *heading* awal 0^o dan diputar searah jarum jam.

Tabel 4. 1 Hasil Pengujian Sensor Kompas HMC5883L

No	Kompas Acuan	HMC5883L	*Error* %
1	0	0.2	2%
2	5	4.7	6%
3	10	6.18	38%
4	15	13.88	8%
5	20	15.15	32%
6	25	24.15	4%
7	30	27.1	11%
8	45	44.15	2%
9	70	68.3	2%
10	90	90.12	0%
11	110	110.77	1%
12	135	135.46	0%
13	160	161.14	1%
14	180	182.23	1%
15	200	204.1	2%
16	225	230.33	2%
17	250	263.78	5%
18	270	274.53	2%
19	290	295.14	2%
20	315	318.77	1%
21	330	332.1	1%
22	345	338.12	2%
23	350	342.33	2%
24	360	357	1%



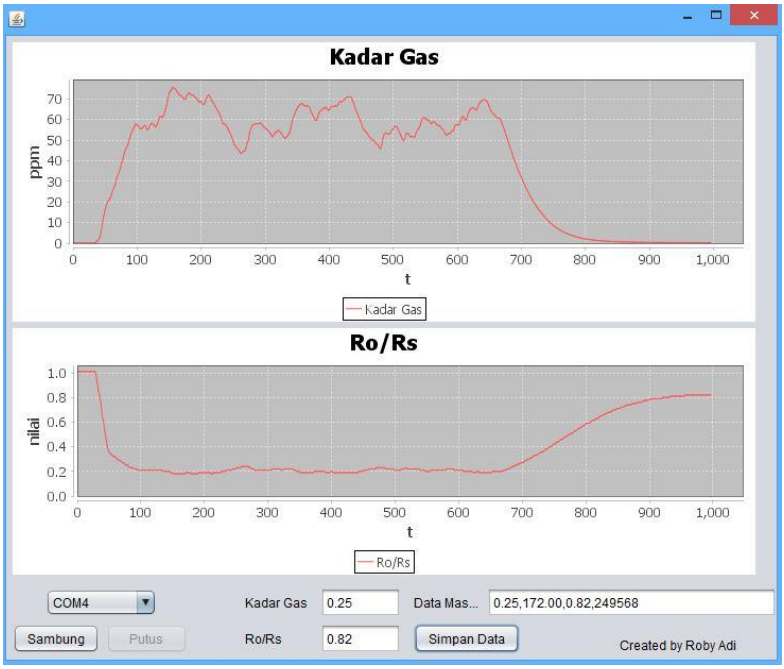
Gambar 4.1 Grafik nilai linier sensor kompas HMC5883L

Dari Tabel 4.1 dapat dilihat bahwa *error* tiap sudut antara hasil pengukuran kompas acuan dengan hasil pengukuran kompas HC5883L memiliki presentasi *error* yang kecil. Kecuali dalam beberapa titik sudut yang menunjukkan *error* cukup besar pada sudut 10º, 20º, dan 30º. Ini dapat disebabkan karena pada sudut-sudut tersebut terdapat benda-benda logam dan medan magnit pengganggu di sekitar sensor kompas sehingga dapat mempengaruhi hasil pembacaan. Pada Gambar 4.1 menunjukkan hasil dari sensor kompas mendekati nilai linier, ini dibuktikan dengan hasil persamaan linier dengan nilai x dan konstanta yang kurang dari 5.

4.2 Pengujian Sensor Gas

Pengujian gas di sini adalah menguji sensor gas jika udara di sekitar sensor diberi polutan. Polutan yang digunakan untuk pengujian ini berupa gas Iso-Butane. Dengan jarak sumber gas terhadap sensor 15 cm, suhu ruangan 25º C, kecepatan kipas *Air Conditioner* pada skala 3 (Panasonic ECO Smart ruang 402). Penggambaran percobaan ditunjukkan pada Gambar 4.2.

Gambar 4. 2 Proses uji sensor gas TGS2602 menggunakan gas *Iso-Butane*



Gambar 4. 3 Hasil pengujian sensor gas

Ppm merupakan kadar gas yang dideteksi, t merupakan waktu dalam satuan milidetik. Dapat dilihat dari grafik Gambar 4.3, dari t saat 100 ms hingga t saat 650 ms, pembacaan gas tidak stabil. Ini dikarenakan pengaruh angin yang dihembuskan oleh AC, menyebabkan arah aliran gas tidak beraturan.

4.3 Pengujian GPS

Pengujian GPS dilakukan dengan membandingkan nilai GPS acuan dengan nilai GPS yang dihasilkan oleh sensor GPS. Pengujian mengambil beberapa sampel lokasi berbeda dalam komplek halaman Gedung Robotika Institut Teknologi Sepuluh Nopember (ITS).

Tabel 4. 2 Pengujian Sensor GPS dengan GPS Acuan

No	Sensor GPS		GPS Ponsel		*Error*		
	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Selisih (Meter)
1	-7.277986	112.798141	-7.277975	112.798143	0.000151%	0.00000177%	1.24
2	-7.277885	112.797996	-7.277878	112.797935	0.000096%	0.00005408%	6.79
3	-7.277818	112.797767	-7.277825	112.797772	0.000096%	0.00000443%	0.95
4	-7.277929	112.797409	-7.277925	112.797412	0.000055%	0.00000266%	0.55
5	-7.278057	112.797508	-7.278055	112.797528	0.000027%	0.00001773%	2.22
6	-7.278006	112.797637	-7.278012	112.79763	0.000082%	0.00000621%	1.02
7	-7.278046	112.797866	-7.278043	112.797857	0.000041%	0.00000798%	1.05

Tabel 4.2 menunjukkan Selisih antara sensor GPS dengan GPS acuan cukup kecil, kecuali pada nomor 2. Selisih terlihat cukup besar mencapai 6 meter. Ini dapat disebabkan karena faktor lingkungan. Di mana saat pengujian, langit tertutup awan mendung pekat. Sehingga sinyal satelit tidak diterima sensor GPS dengan baik.

4.4 Pengujian Kendali PWM terhadap Error Sudut

Nilai *error* didapatkan dari selisih antara *setpoint* dengan nilai *heading*. Nilai *error* ini kemudian dimasukkan dalam persamaan kendali PID, di mana nilai proposional sebagai *gain* yang didapat dari perkalian nilai *error* dengan kP. Nilai integral didapatkan dari penjumlahan nilai *error* tiap waktu yang dikalikan dengan kI. Dan nilai *derivative*

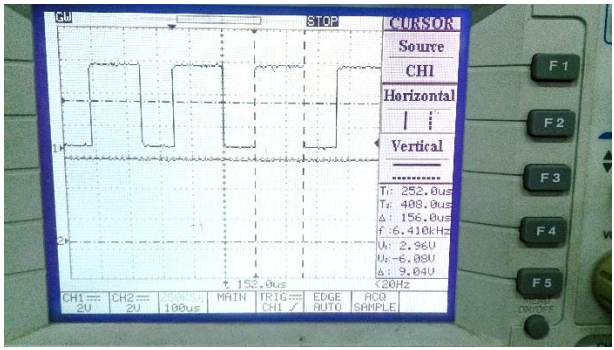
didapatkan dari selisih *error* tiap waktu dikalikan dengan kD untuk menghasilkan nilai pengaturan PWM.

Pada pengujian ini nilai *setpoint* diatur pada 0º, kemudian robot diputar ke arah kanan dan ke arah kiri secara bergantian. *Error* bernilai negatif ketika *heading* diarahkan ke kanan, dan bernilai positif ketika diarahkan ke kiri terhadap nilai *setpoint*.

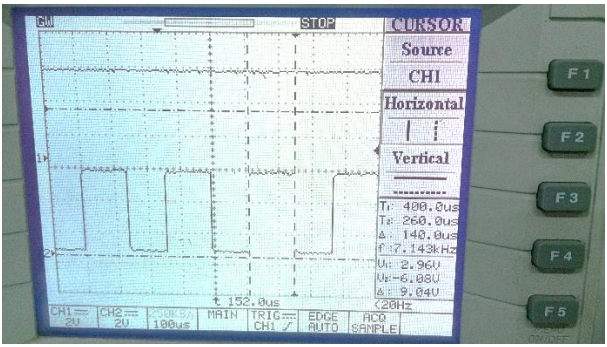
Ketika *heading* diarahkan ke kiri, sudut bergeser dari 0º hingga 25º. Nilai PWM bagian kiri akan tetap pada 255, dan nilai PWM kanan mengalami penurunan seperti yang ditunjukkan Gambar 4.4. Semakin besar nilai error yang dihasilkan, maka semakin besar penurunan nilai PWM bagian kanan.

a. Robot Diarahkan ke Kanan
   Error: -25
   PWM_ka: 255
   PWM_ki: 140
   Duty cycle = T_{on} / T_{off} = 156/256 = 0,61

Ketika *heading* diarahkan ke kanan, sudut bergeser dari 0º hingga 25º. Nilai PWM bagian kanan akan tetap pada 255, dan nilai PWM kiri mengalami penurunan seperti yang ditunjukkan Gambar 4.5. Semakin besar nilai error yang dihasilkan, maka semakin besar penurunan nilai PWM bagian kiri.



Gambar 4.4 PWM ketika error -25



Gambar 4.5 Nilai PWM ketika error 25

b. Robot Diarahkan ke Kiri
Error: 25
PWM_ka: 140
PWM_ki: 255
Duty cycle = T_{on} / T_{off} = 140/256 = 0,54

Nilai *error* yang dihasilkan dari pergeseran sudut antara *setpoint* dan heading merubah nilai PWM. Nilai PWM ini akan mengendalikan kecepatan putaran roda robot. Dengan menggunakan prinsip *differential speed steering*, robot akan dapat bermanuver mengikuti arah *setpoint angle* dengan memanfaatkan perbedaan kecepatan putaran roda yang dihasilkan dari perbedaan nilai PWM.

4.5 Pengujian Navigasi Waypoint

Pengujian navigasi *waypoint* dilakukan untuk mengetahui seberapa besar ketepatan robot dalam bernavigasi menuju titik-titik *waypoint* yang ditentukan.

Tabel 4. 3 Daftar Titik Waypoint

No WP	*Latitude*	*Longitude*
1	-7.277986	112.798141
2	-7.277885	112.797996
3	-7.277818	112.797767
4	-7.277929	112.797409

No WP	*Latitude*	*Longitude*
5	-7.278057	112.797508
6	-7.278006	112.797637
7	-7.278046	112.797866

Titik-titik *waypoint* yang ditunjukkan pada tabel 4.3, selanjutnya di-*plot* pada aplikasi peta untuk menunjukkan titik-titik pada peta dan menggambar garis yang menghubungkan antar titiknya. Hasil *plotting* dari *waypoint* ditunjukkan Gambar 4.6.

Gambar 4.6 Hasil plot titik waypoint

Setelah *waypoint* dimasukkan, robot dijalankan dan mengirimkan koordinat perjalanan setiap 5 detik. Hasil data pengiriman koordinat ditunjukkan pada tabel 4.4

Tabel 4. 4 Data Koordinat Perjalanan

No	Latitude	Longitude
1	-7.277979	112.79811
2	-7.277979	112.79811
3	-7.277962	112.79808
4	-7.277945	112.798065
5	-7.277945	112.798065
6	-7.277923	112.798011
7	-7.277868	112.79795
8	-7.277854	112.797897
9	-7.277854	112.797866
10	-7.277854	112.797866
11	-7.277849	112.797767
12	-7.277852	112.797706

No	Latitude	Longitude
13	-7.277865	112.797676
14	-7.277879	112.797607
15	-7.277883	112.797607
16	-7.277892	112.797523
17	-7.277887	112.797477
18	-7.277875	112.797462
19	-7.277883	112.797462
20	-7.277913	112.797409
21	-7.277926	112.797393
22	-7.277949	112.797409
23	-7.277984	112.797447
24	-7.278026	112.797462

Gambar 4.7 Hasil *plotting* jalur waypoint dengan jalur yang dilalui oleh robot

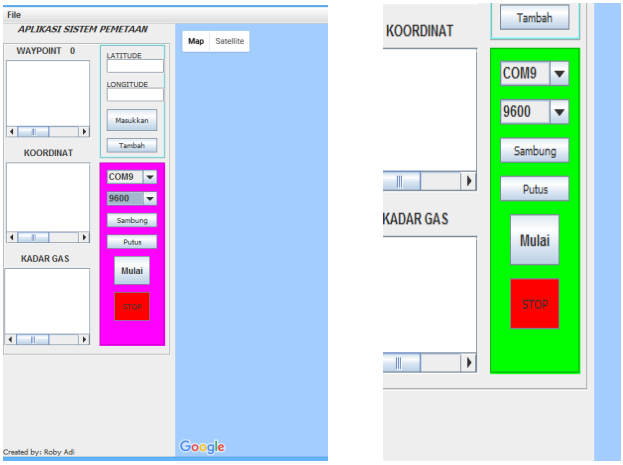
Data hasil perjalanan dan data *waypoint* kemudian di-*plot* seperti pada Gambar 4.7 untuk menunjukkan jalur yang telah dilalui oleh robot dengan jalur yang seharusnya dilalui.

Garis hijau merupakan garis perjalanan, sedangkan garis merah merupakan garis jalur *waypoint.* Pada beberapa titik terjadi simpangan antara jalur yang seharusnya dengan jalur yang dilewati oleh robot. pada titik 1 pergeseran sebesar 2,91 meter, pada titik 2 pergeseran sebesar 3,5 meter, pada titik 3 pergeseran sebesar 3,81 meter. Ini dapat dikarenakan simpangan sensor kompas. Kemudian jumlah satelit yang mengunci GPS juga sangat berpengaruh. Semakin banyak jumlah satelit yang mengunci, semakin tinggi keakuratan dari GPS.

4.6 Pengujian Aplikasi Pemetaan

Pengujian ini meliputi koneksi antara robot dengan program aplikasi, pengujian transmisi data antara robot dengan program aplikasi, menampilkan data *waypoint*, dan menampilkan data titik perjalanan robot.

4.6.1 Pengujian Koneksi Serial



(a) (b)

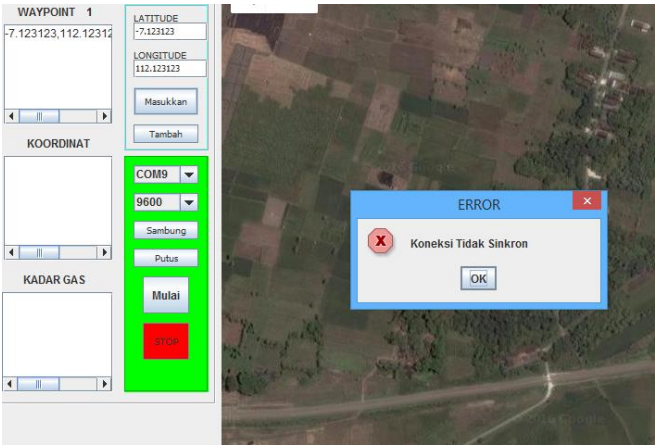
Gambar 4.8 Koneksi antara aplikasi dengan robot (a: not connected; b: connected)

Pengujian ini dilakukan dengan menghubungkan aplikasi dengan robot melalui koneksi *serial comport.* Saat belum terkoneksi (Gambar

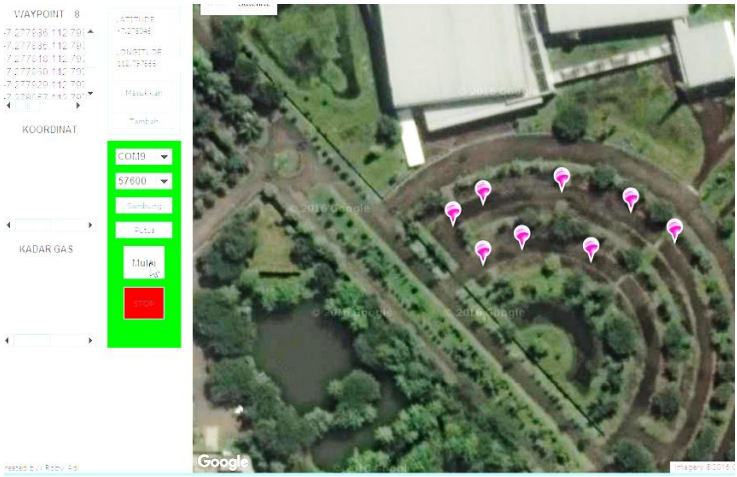
4.8a) panel background berwarna magenta. Ketika koneksi antara aplikasi program dengan robot berhasil (Gambar 4.8b), maka warna *background* panel koneksi akan berubah menjadi warna hijau. Dan ketika tidak berhasil, maka akan muncul peringatan bahwa koneksi tidak berhasil. Koneksi tidak berhasil dapat dikarenakan jalur *comport* yang tidak sama, dapat juga dikarenakan nilai *baudrate* yang tidak sinkron.

4.6.2 Pengujian Transmisi Data

Pengujian berupa pengiriman data dari program aplikasi kepada robot, dan juga sebaliknya, berupa data koordinat *waypoint*. Pengiriman data ini dapat juga mengalami kegagalan (Gambar 4.9). Dapat dikarenakan *lost connection* dapat juga dikarenakan kegagalan sistem aplikasi karena membaca data yang tidak sinkron.



Gambar 4. 9 Transfer data tidak berhasil



Gambar 4.10 Tampilan waypoint

4.6.3 Pengujian Menampilkan Lokasi *Waypoint*

Pengujian ini dilakukan dengan memasukkan nilai-nilai *latitude* dan *longitude* pada *textfield latitude* dan *longitude* di aplikasi (Gambar 4.10). Jumlah titik *waypoint* yang dapat diberikan pada aplikasi ini dibatasi hanya berjumlah 100 *waypoint* saja. Ini dikarenakan jumlah *memory* data yang dikirimkan tidak membuat penuh memori robot.

4.6.4 Pengujian Menampilkan Titik Lokasi Perjalanan

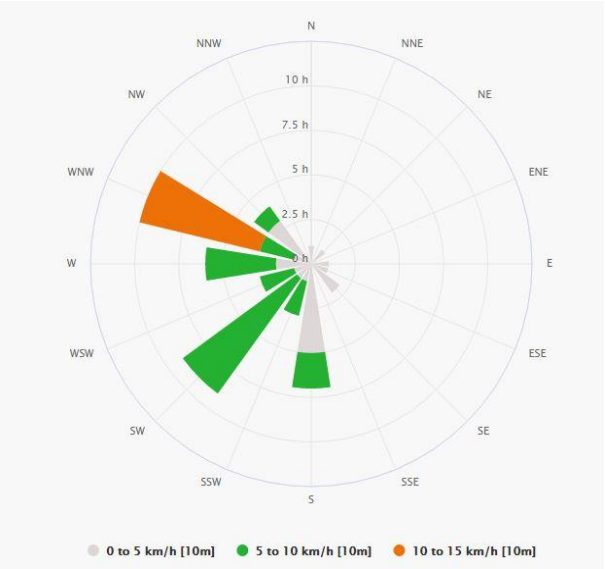
Pengujian ini dilakukan ketika robot melakukan perjalanan ke *waypoint*. Karena selama itu, robot mengirimkan data koordinat perjalanannya pada aplikasi (Gambar 4.11). Data itu berupa deretan *string* dengan format $a,nilai_Latitude,nilai_Longitude,informasi.

Data informasi akan secara langsung ditampilkan pada kotak dialog koordinat perjalanan. Sehingga secara langsung, kita dapat mengetahui informasi apa yang dikirim oleh robot kepada kita melalui aplikasi ini. Kecepatan aplikasi peta ketika memuat data untuk ditampilkan antara 0,5 detik hingga 2 detik. Ini tergantung dari kecepatan internet yang digunakan oleh server.

Gambar 4.11 Tampilan titik perjalanan

4.7 Pengujian Integrasi Sistem

Pengujian ini adalah pengujian yang dilakukan pada sistem yang sudah terintegrasi untuk kemudian diuji pada lapangan yang sebenarnya. Pengujian ini dilakukan di Lapangan Pertamina. Dengan memasukkan tiga titik *waypoint*. *Waypoint* 1 pada lokasi -7.284085;112.792625, lokasi *waypoint* 2 pada -7.284270;112.792854, dan *waypoint* 3 pada lokasi -7.284009;112.792922. Sumber gas diletakkan pada dua titik yang berlokasi di *waypint* 1 dan *waypoint* 2. Hasil pemetaan dari pengujian integrasi sistem ini ditunjukkan pada Gambar 4.13



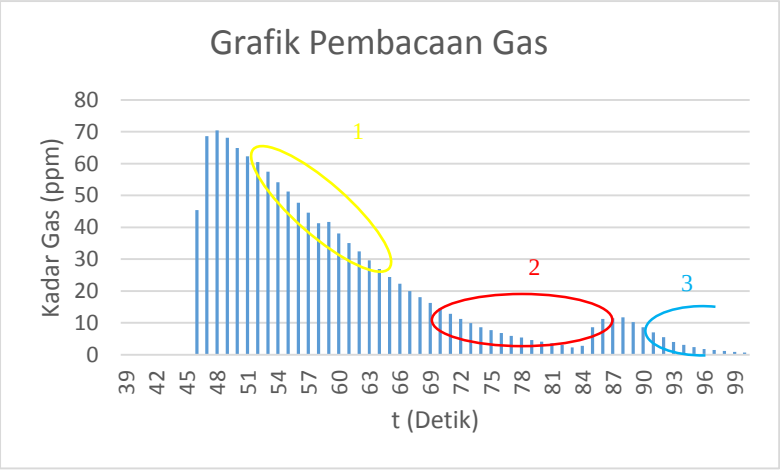
Gambar 4. 12 Arah dan kecepatan angin

Gambar 4. 13 Hasil peta pengujian integrasi sistem

Dari titik awal keberangkatan menuju titik 1, kecepatan tertinggi angin mengarah diantara barat dan barat laut (Gambar 4.12). Sehingga arah penyebaran gas mengikuti arah angin antara barat dan barat laut. Terlihat bahwa ketika robot mulai mendekati titik 1, di mana sumber gas lokasi 1 diletakkan, grafik pembacaan gas (Gambar 4.14) mengalami peningkatan kadar ppm. Pada peta (Gambar 4.13) ditampilkan nilai ppm sebesar 38, ini dikarenakan periode penampilan data ke peta setiap 20 detik, sehingga data yang tampil ketika t = 60 = 38 ppm.

Kemudian robot menuju titik 2, dimana robot menjauhi sumber gas lokasi 1. Dan grafik menunjukkan nilai penurunan ppm. Saat t = 80, robot belum mencapai titik 2, tetapi mulai mendeteksi kadar gas 4 ppm. Mendekati titik 2 sebaagai sumber gas kedua, grafik nilai ppm naik. Tetapi tidak sebesar saat berada pada titik 1. Ini dikarenakan hanya sumber gas 2 saja yang mempengaruhi nilai ppm. Berbeda dengan ketika berada pada lokasi 1, dimana sumber gas 1 dan sumber gas lokasi 2 mempengaruhi nilai ppm karena angin yang mengarah diantara barat dan barat daya.

Selanjutnya, robot mengarah pada titik lokasi 3. Grafik ppm gas menurun. Karena robot menjauhi sumber gas lokasi 2 dan tidak terdapat sumber gas pada titik 3.



Gambar 4. 14 Data gas lokasi pengujian

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan data hasil uji implementasi *Autonomous Navigation Robot* ini diperoleh beberapa kesimpulan antara lain dua komponen penting dalam sistem navigasi otomatis yaitu kompas dan GPS sangat berpengaruh terhadap ketepatan rute perjalanan robot. Ketika simpangan kompas melebihi 30%, dapat mengakibatkan arah heading tidak berada pada nilai *setpoint* yang benar. Ketidakakuratan dari GPS berpengaruh pada kebenaran nilai *setpoint* yang dihasilkan.

Kecepatan dan arah angin dapat mempengaruhi nilai pembacaan sensor gas. Ketika angin mengenai objek uji, pembacaan nilai kadar gas mengalami penurunan, kemudian mengalami kenaikan lagi ketika objek uji tidak terkena angin.

Penggunaan *Autonomous Navigation Robot* yang dilengkapi dengan sensor GPS Ublox Neo M8 dan sensor gas TGS2600 dapat memetakan kadar gas suatu lokasi atau daerah dengan baik. Menghasilkan simpangan jarak kurang dari 5 meter dalam mengikuti rute waypoint, memberikan informasi koordinat yang akurat dengan radius kurang dari 5 meter dan memberikan informasi kenaikan dan penurunan kadar gas kepada server selama perjalanan sebagai sumber informasi peta

5.2 Saran

Penambahan sensor Inertia Measurement Unit (IMU) untuk meningkatkan keakuratan sistem GPS hingga 1 meter. Penambahan jumlah sensor gas pada empat sisi robot, untuk pendeteksian gas dari segala arah, untuk menambah keakuratan pembacaan gas hingga 0,01 ppm pada lokasi yang dipetakan.

Halaman Ini Sengaja Dikosongkan

DAFTAR PUSTAKA

[1] Sika.pom.go.id/v2015/artikel/GAS%BERACUN.pdf, 11 Januari 2016

[2] Prawiro, Ruslan H. 1988. Ekologi Pencemaran Lingkungan, Satya Wacana, Semarang

[3] ___**,** Figaro**.**<URL: http://www.figaro.co.jp/en/product/> 12 Juni 2016.

[4] Ikhsan, Muhammad Anshori, "Desain Kontrol Autopilot pada UGV (Unmanned Ground Vehicle) Berbasis GPS (Global Positioning System), Teknik Elektro Universitas Diponegoro, 2011.

[5] Rengarajan M., Anitha G., "Algorithm Development and Testing of Low Cost Waypoint Navigation System". ISSN: 2250-3498, Vol.3, No.2, April 2013.

[6] Goge, Douglas W., "A Brief History of Unmanned Ground Vehicle (UGV) Development Efforts", Unmanned System Magazine, United States of America, 1995.

[7] Stefan Jeff, Navigating with GPS, Circuit Cellar Magazine, USA, 2000.

[8] ___, <ayomaonline.com/iot>, 15 Januari 2017

[9] ___, Ublox Neo M8 Datasheet,<URL https://www.u-blox.com/>, 11 Januari 2016.

[10] Carusso, Michael J. *"*Application of Magnetoresistive Sensors in Navigation Systems*"*. SAE SP-1220, 15-21, Feb. 1997.

[11] ___, HMC5883L Datasheet,<URL: www.infineon.com/../BTN7960B/> 18 Juni 2016.

[12] ___, <www.amazon.com/hmc5883l>, 15 Januari 2017

[13] ___, <arduino-info.wikispaces.com>, 15 Januari 2017

[14] ___, <www.sparkfun.com/library/arduino-mega-pinout>, 15 Januari 2017

[15] Wu Xiaodong, Min Xu, Lei Wang. "Differential Speed Steering Control for Four-Wheel Independent Driving Electric Vehicle". International Journal of Materials, Mechanics and Manufacturing, Vol. 1, No. 4, November 2013.

[16] Ogata, katsuhiko, Edi Laksono (Penterjemah). 1993. Teknik Kontrol Automatik (Sistem Pengaturan) Jilid 2. Jakarta : Erlangga.

[17] __, BTN7960 Datasheet,<URL: www.infineon.com/../BTN7960B/> 20 Juni 2016.

[18] __, <http://bigtronica.com/drivers/335-driver-puente-h-40a-bts7960-5053212003357.html>, 15 Januari 2017

[19] __, <id.aliexspress.com/3dr>, 15 Januari 2017

[20] __, <www.ardupilot.org/connection3dr>, 15 Januari 2017

[21] Ichtiara Cita, 2008 "*Implementasi Aplikasi Sistem Informasi Geografis Universitas Indonesia Berbasis Web Dengan Menggunakan Google Maps API*". Universitas Indonesia.

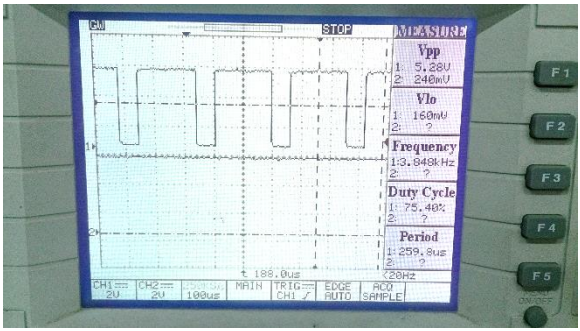
[22] _, Google Maps API,<URLhttps://developers.google.com/maps/>, 12 September 2016.

[23] ___, **NetBeans**,<URL: http://www.Netbeans.org>, 27 September 2016.

LAMPIRAN

A. Uji PWM

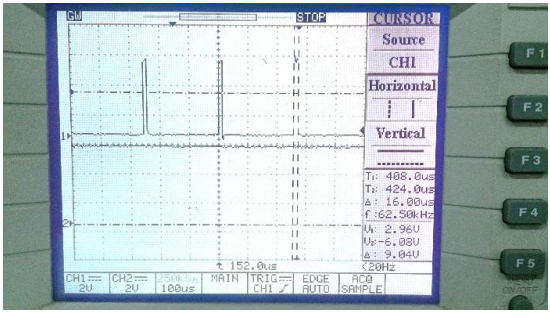
- *Error*: -15
  PWM_ka: 255
  PWM_ki: 192



Gambar 1 Duty cycle *error* -15

Duty cycle = T_{on} / T_{off} = 192/256 = 0,75

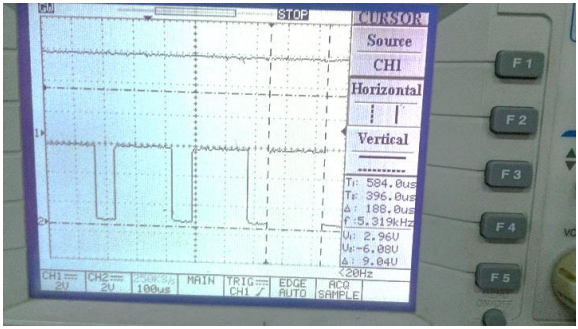
- *Error*: -50
  PWM_ka: 255
  PWM_ki: 11



Gambar 2 Duty cycle *error* -50

Duty cycle = T_{on} / T_{off} = 16/256 = 0,06

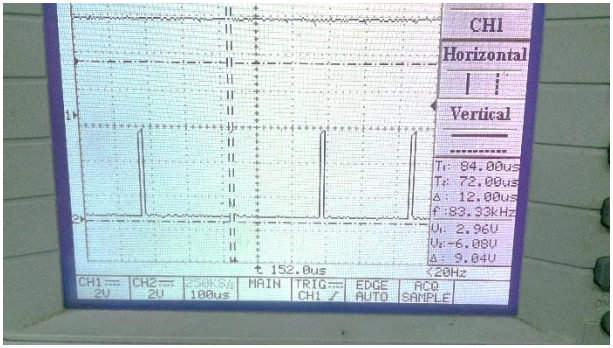
- *Error*: 15
  PWM_ka: 185
  PWM_ki: 255



Gambar 3 Duty cycle *error* 15

Duty cycle = T_{on} / T_{off} = 188/256 = 0,73

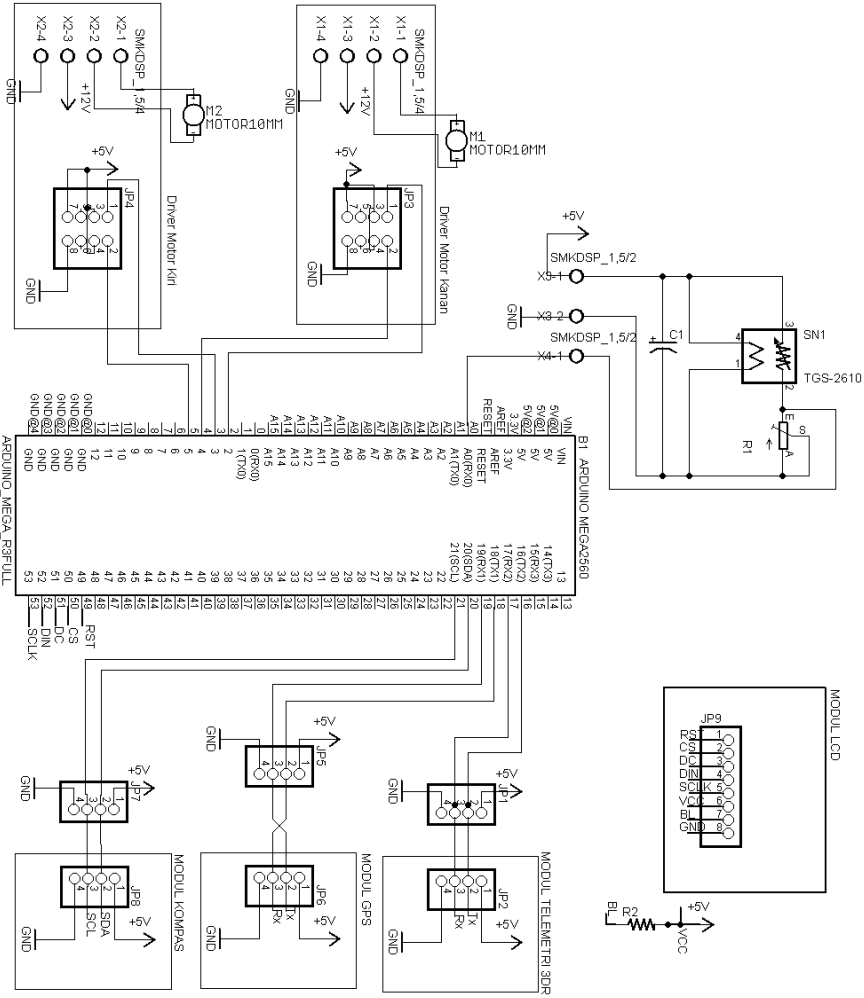
- *Error*: 50
  PWM_ka: 15
  PWM_ki: 255



Gambar 4 Duty cycle *error* 50

Duty cycle = T_{on} / T_{off} = 12/256 = 0,046

B. Gambar Skematik Rangkaian Keseluruhan



C. Source Code Robot

```
#define _TASK_TIMECRITICAL
#define _TASK_PRIORITY
#include <TaskScheduler.h>
//#include <DirectIO.h>
//#include <PinChangeInt.h>

#include <NewPing.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <Arduino.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <SPI.h>
#include <Wire.h>

#include <Ublox.h>
#include "compass.h"

Ublox M8_GPS;

#define TRIGGERPIN      45
#define ECHOPIN         46
#define MAX_DISTANCE    100

#define PIN_ADC         0
#define PIN_R_MOTOR_KA  2
#define PIN_L_MOTOR_KA  4
#define PIN_R_MOTOR_KI  3
#define PIN_L_MOTOR_KI  5

#define SCLK            53
#define DIN             52
#define DC              51
#define CS              50
#define RST             49

//#define _DEBUG_

#define Task_t          10

Adafruit_PCD8544 display = Adafruit_PCD8544(SCLK, DIN, DC,
CS, RST);
NewPing sonar(TRIGGERPIN, ECHOPIN, MAX_DISTANCE);

boolean error;
```

```
#define PING_PERIODE            50  //50 kali per detik
#define KOMPAS_PERIODE          10  //100 kali per detik
#define GPS_PERIODE             20 //5 kali per detik
#define GAS_PERIODE             1000 //1 kali per detik
#define PID_PERIODE             5   //200 kali per detik
#define WAYPOINT_PERIODE        250  //4 kali per detik
#define JARAK_ANGLE_PERIODE     125  // 6 kali per detik
#define KIRIMDATA_PERIODE       20000//10 detik sekali
#define TAMPIL_PERIODE          300  //3 kali per detik
#define TERIMADATA_PERIODE      50  //20 kali per detik

#define PWM_MAX           255

//Callback methode
void pingCallback();
void kompasCallback();
void gpsCallback();
void gasCallback();
void pidCallback();
void waypointCallback();
void jarakAngleCallback();
void kirimDataCallback();
void tampilCallback();
void terimaDataCallback();
//void kalCallback();

//Tasks
//parameterKendali_Manager prioritas lebih tinggi
//tugas_Manager prioritas umum
Scheduler tugasManager, tugasAnakBuah;

Task tUltrasound(PING_PERIODE, TASK_FOREVER, &pingCallback);
Task tKompas(KOMPAS_PERIODE, TASK_FOREVER, &kompasCallback);
Task tGPS(GPS_PERIODE, TASK_FOREVER, &gpsCallback);
Task tGas(GAS_PERIODE, TASK_FOREVER, &gasCallback);
Task tPID(PID_PERIODE, TASK_FOREVER, &pidCallback);
Task tWP(WAYPOINT_PERIODE, TASK_FOREVER, &waypointCallback);
Task tJarakAngle(JARAK_ANGLE_PERIODE, TASK_FOREVER,
&jarakAngleCallback);
Task tKirimData(KIRIMDATA_PERIODE, TASK_FOREVER,
&kirimDataCallback);
Task tTampil(TAMPIL_PERIODE, TASK_FOREVER, &tampilCallback);
Task tTerima(TERIMADATA_PERIODE, TASK_FOREVER,
&terimaDataCallback);
//Task tKalGas(2, TASK_ONCE, &kalCallback);

/** Variabel **/
int x,y,z,i,j,k,n;

//PING
  float jarak;
```

```
//Kompas
  int heading = 0;

//GPS
  double lat_skrg, lon_skrg;
  int ketinggian, satelit;

//GAS
  double kadarGas, nilaiAwalADC, Ro;

//PID
  int Error;
  float PID;
  int nilaiPWM_ka;
  int nilaiPWM_ki;

//Waypoint
  int jumlahTujuan, tujuan, m;
  double lat[100];
  double lon[100];
  double lati[100];
  double loni[100];

//Hitung Angle Jarak
  int hasilJarakTitik;
  int Angle;

//Terima Data
  uint8_t GPSindex = 0;
  char GPSbuff[80];

/************************  AMBIL NILAI RO
*************************************/
void kalCallback(){
  //Serial2.println("Kalibrasi Bro");
  nilaiAwalADC = analogRead(PIN_ADC);
  float tegangan = nilaiAwalADC * (5.0/1023.0);
  Ro = ((4.9 * 10000.0)/tegangan) - 10000;
}

/*****************************************************/
/************************  PING
*************************************/
void pingCallback(){
  float xx = sonar.ping_cm();
  #ifdef _DEBUG_
   Serial.println("Ping = " + String(millis()) + ",  " +
String(xx));
  #endif
  }
```

```
/*******************************************************/
/************************  KOMPAS
*****************************/
void kompasCallback(){
    int dt;
    unsigned long t;
    float load;

    t = millis();
    compass_scalled_reading();
    compass_heading();
    dt = millis() - t;
    load = dt/(Task_t/100);
    heading = bearing;

    #ifdef _DEBUG_
     Serial.println("Kompas = " + String(millis()));
    #endif
  }
/*******************************************************/

/************************** GPS
*****************************/
void gpsCallback(){
    while(Serial1.available() > 0){
      char c = Serial1.read();
      if(M8_GPS.encode(c)){
          ketinggian = M8_GPS.altitude;
          lat_skrg   = M8_GPS.latitude;
          lon_skrg   = M8_GPS.longitude;
          satelit    = M8_GPS.sats_in_use;
        }
      }
      Serial.println(lat_skrg);
      Serial.println(lon_skrg);Serial.println(satelit);
     #ifdef _DEBUG_
     Serial.println("GPS = " + String(millis()));
     #endif
  }
/*******************************************************/

/************************  GAS
*****************************/
void gasCallback(){
  ADC = analogRead(PIN_ADC);
  float const eksponen = 2.718282;
  float batas = nilaiAwalADC;
  float level = map(ADC, batas, 1023, 0, 100); //kadar
persen

  float tegangan = ADC * (5.0/1023.0);
```

```
  float Rs = (4.9 * 10000.0)/tegangan;
  Rs = Rs - 10000.0;

  float perbandingan = Rs/Ro;
  float nilai = (-8.868) * perbandingan;
  float nilaiEksponen = pow(eksponen, nilai);
  kadarGas = 366.84 * nilaiEksponen;
  if(kadarGas < 0){kadarGas = 0;}
  //Serial.println("Gas = " + String(kadarGas) + ", ADC = "
+ String(nilaiAwalADC) + ", Ro/Rs = " + String(perbandingan)
+ ", Millis = " + String(millis()));

  Serial2.println("$kG," + String(kadarGas) + "," +
String(nilaiAwalADC) + "," + String(perbandingan) + "," +
String(millis()));

  #ifdef _DEBUG_
  Serial.println(String(kadarGas) + "," +
String(nilaiAwalADC) + "," + String(perbandingan) + "," +
String(millis()));
  #endif
  }
/*******************************************************/
/************************  PID
*****************************/
void pidCallback(){
  float kP, kI, kD;
  float Dt;

  int lastError;
  unsigned long waktuSekarang;
  unsigned long waktuLalu;

  kP = 3;
  kI = 0.000035;
  kD = 5.2;

  heading = bearing;
  Error = Angle - heading;

  if (Error > 180) {
      Error = -1 * (360.00 - (Angle - heading));
  }
  else if (Error < -180) {
      Error = (360.00 + (Angle - heading));
  }

  waktuSekarang = millis();
  Dt = (float)(waktuSekarang - waktuLalu); //Delta waktu,
waktuSkrg - waktuYgLalu
```

```
  float P = Error * kP;
  float I = ((Error*Dt) + I) * kI;
  float D = ((Error - lastError) * kD)/Dt;

  PID = P + I + D;
  lastError = Error;
  //Serial.println("Error=" + String(Error) + " " +
"LastError=" + String(lastError) + " " + "PID=" +
String(jambu));
  waktuLalu = waktuSekarang; //jadi waktuYgLalu

   nilaiPWM_ka = (255 - PID);
   nilaiPWM_ki = (255 + PID);

  if(nilaiPWM_ka > PWM_MAX){
      nilaiPWM_ka = PWM_MAX;
        if (nilaiPWM_ki < 0){
            kiriTajam();
          }
       else {
          maju();
        }
    }

  if(nilaiPWM_ki > PWM_MAX){
      nilaiPWM_ki = PWM_MAX;
        if(nilaiPWM_ka < 0){
            kananTajam();
          }
         else{
            maju();
        }
    }
  #ifdef _DEBUG_
  Serial.println("PID = " + String(millis()));
  #endif
}

void maju(){
  analogWrite(PIN_R_MOTOR_KA, nilaiPWM_ka);
  analogWrite(PIN_L_MOTOR_KA, 0);

  analogWrite(PIN_R_MOTOR_KI, nilaiPWM_ki);
  analogWrite(PIN_L_MOTOR_KI, 0);
  }

void kiriTajam(){
  analogWrite(PIN_R_MOTOR_KA, nilaiPWM_ka);
  analogWrite(PIN_L_MOTOR_KA, 0);

  analogWrite(PIN_R_MOTOR_KI, 0);
```

```
  analogWrite(PIN_L_MOTOR_KI, nilaiPWM_ki);
}

void kananTajam(){
  analogWrite(PIN_R_MOTOR_KA, 0);
  analogWrite(PIN_L_MOTOR_KA,nilaiPWM_ka);

  analogWrite(PIN_R_MOTOR_KI, nilaiPWM_ki);
  analogWrite(PIN_L_MOTOR_KI, 0);
}

/*******************************************************/
/************************  WAYPOINT
**********************/
void waypointCallback(){
    lat[tujuan] = lati[m];
    lon[tujuan] = loni[m];
      if (hasilJarakTitik > 0 && hasilJarakTitik < 2){
        //hasilJarakTitik = 0;
        tujuan += 1;
        m += 1;
        jumlahTujuan -= 1; //Jika tujuan sudah habis. maka
berhenti
        if (jumlahTujuan == 0)
        {
            tPID.disable();
            tKirimData.disable();
            //tugasManager.disableAll(true);
            berhenti();
        }
       }
     #ifdef _DEBUG_
      Serial.println("WP = " + String(millis()));
     #endif
  }
/*******************************************************/

/************************  JARAK & ANGLE
********************/
void jarakAngleCallback(){
  float hitungJarakTitik = sqrt(((lon_skrg - lon[tujuan]) *
(lon_skrg - lon[tujuan])) + ((lat[tujuan] - lat_skrg) *
(lat[tujuan] - lat_skrg)));
        hitungJarakTitik *= 110567; //Satuan ubahan ke meter
        hasilJarakTitik = hitungJarakTitik;
                                    // (Ytarget   /
Xtarget)
  float hitungSudut = atan2((lon[tujuan] - lon_skrg),
(lat[tujuan] - lat_skrg));//sudul alfa, arah angle set point
  float rubahKeRadian = 57.29577951; //ubah radian ke sudut
        hitungSudut *= rubahKeRadian;
```

```
      if (hitungSudut < 0){
       hitungSudut = 360 + hitungSudut;
       }
      else if (hitungSudut > 0){
       hitungSudut = hitungSudut;
       }
      else if (hitungSudut > 360){
       hitungSudut = 0;
       }

      Angle = hitungSudut;
      #ifdef _DEBUG_
       Serial.println("Jarak & Angle = " + String(millis()));
      #endif
  }
/*********************************************************/

/************************  KIRIM DATA
**********************/
void kirimDataCallback(){
    Serial2.println("$a," + String(lat_skrg,6) + "," +
String(lon_skrg,6) + "," + String(kadarGas));
    #ifdef _DEBUG_
      Serial.println("Kirim Data = " + String(millis()));
    #endif
  }
/*********************************************************/

/************************  TAMPIL
************************/
void tampilCallback(){
  display.setTextSize(0.5);
  display.setTextColor(BLACK);
  display.setContrast(30);
  display.setCursor(0,0);
  display.print("Head=");
  display.setCursor(38,0);
  display.println(heading);
  display.setCursor(0,8);
  display.print("Angle=");
  display.setCursor(38,8);
  display.println(Angle);

  //display.display();
  //display.clearDisplay();
  display.setCursor(0,18);
  display.print("Lt=");
  display.println(lat_skrg, 6);
  display.setCursor(0,26);
  display.print("Ln=");
```

```
  display.println(lon_skrg, 6);
  display.setCursor(0,36);
  display.print("St=");
  display.print(satelit);
  display.print("||");
  display.print("Jr=");
  display.print(hasilJarakTitik);
  display.display();
  display.clearDisplay();
  //Serial.println("Tampil = " + String(millis()));
  }
/*******************************************************/

/*********************  TERIMA DATA
**********************/
void terimaDataCallback(){
    char index;

    //Serial.println("Loop");
    while (Serial2.available()) {
     index = Serial2.read();
     if (index == 'f') {
         Serial.println("Berangkat Bos");
         kalCallback();
         tugasAnakBuah.enableAll(true); //task scheduler
diaktifkan
         tugasAnakBuah.execute();
     }
     if (index == 'p'){
        tPID.disable();
        tKirimData.disable();
        berhenti();
      }
     if (index == 's'){
        tPID.disable();
        tKirimData.disable();
        tGas.disable();
        //tKalGas.restart();
        berhenti();
     }
    if(index == '$' || GPSindex >= 80){
    GPSindex = 0;
  }

  if(index != '\r'){
      GPSbuff[GPSindex++] = index;
    }

  if(index == '\n'){
    prosesGPS();
        GPSindex = 0;
```

```
        jumlahTujuan += 1;;
      }
    }
  // Serial.println("TerimaData = " + String(millis()));
  }

void prosesGPS(){
    String inputString1 = "";
    String inputString2 = "";
    char buff1[12];
    char buff2[14];
    char s[1];
    s[0] = GPSbuff[1];
    n = atoi(s);
    //Serial2.println(n);
    for (i = 3, j= 0, k= 0;(i <= GPSindex) && (j < 3); i++)
{
        if(GPSbuff[i] == ','){
            j++;
        }
    else{
      if(j == 0){
          char inChar1 = GPSbuff[i];
        if (GPSbuff[i] >= '0' && GPSbuff[i] <= '12'){
          inputString1 += inChar1;
          inputString1.toCharArray(buff1,13,0);
              }
           }
       if(j == 1){
           char inChar2 = GPSbuff[i];
             if(GPSbuff[i] >= '0' && GPSbuff[i] <= '14'){
                 inputString2 += inChar2;
                   inputString2.toCharArray(buff2, 14, 0);
               }
           }
        }
    }


    lati[n] = atof(buff1);
    lati[n] = (lati[n]/1000000)*(-1);
    loni[n] = atof(buff2);
    loni[n] = (loni[n]/1000000);

    delay(100);
                    //$wp,1,-7.123123,112.123123
    Serial2.println("$wp," + String(n) + "," +
String(lati[n],6) + "," + String(loni[n],6));

//    Serial2.print("$");
//    Serial2.print(n);
```

```
//    Serial2.print(",");
//    Serial2.print(lati[n],6);
//    Serial2.print(",");
//    Serial2.println(loni[n],6);//Serial.print("wp");

    //Serial.println("");
    //n++;
   // Serial.println("Proses GPS = " + String(millis()));
}

void berhenti(){
  analogWrite(PIN_R_MOTOR_KA, 0);
  analogWrite(PIN_L_MOTOR_KA, 0);

  analogWrite(PIN_R_MOTOR_KI, 0);
  analogWrite(PIN_L_MOTOR_KI, 0);
}
/*********************************************************/
/*************************  SETUP
*****************************/
void setup(){
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(19200);
  Wire.begin();
  display.begin();
  display.clearDisplay();

  tugasAnakBuah.addTask(tTampil);
  tugasAnakBuah.addTask(tKirimData);
  tugasAnakBuah.addTask(tWP);
  tugasAnakBuah.addTask(tGas);

  //tugasManager.addTask(tKalGas);
  tugasManager.addTask(tGPS);
  tugasManager.addTask(tUltrasound);
  tugasManager.addTask(tKompas);
  tugasManager.addTask(tJarakAngle);
  tugasManager.addTask(tTerima);
  tugasManager.addTask(tPID);

  tugasAnakBuah.setHighPriorityScheduler(&tugasManager);

  TCCR3B = (TCCR3B & 0xF8)|0x02;

  tujuan = 1;
  m=1;
  kadarGas = 0;
  n = 0;
  jumlahTujuan = 0;
```

```
  tTampil.enable();
  tGPS.enable();
  tKompas.enable();
  tTerima.enable();
  //tGas.enable();

  //tugasAnakBuah.enableAll(true);
  //tugasManager.enableAll(true);
  }
/*******************************************************/
/*************************  LOOP
*****************************/
void loop(){
    tugasAnakBuah.execute();
    //tugasManager.execute();
  }
```

BIODATA PENULIS

Roby Adi Wibowo. dilahirkan di Grobogan, pada tanggal 11 Februari 1992 merupakan putra kelima dari empat bersaudara pasangan Bapak Giyanto dan Ibu Mustiah. Penulis menamatkan sekolah di SDN 16 Purwodadi tahun 2004. Kemudian masuk ke SMPN 1 Purwodadi, tamat tahun 2007, dan melanjutkan di SMKN 2 Purwodadi pada tahun 2010. Tahun 2010, penulis melanjutkan pendidikan di D3 Teknik Elektro Fakultas Teknik Universitas Diponegoro Semarang dan tamat pada tahun 2013. Selanjutnya penulis mengambil pendidikan S1 program Lintas Jalur Teknik Elektro, Fakultas Teknik Iindustri – Institut Teknologi Sepuluh Nopember Surabaya pada akhir tahun 2013. Penulis memilih bidang studi Elektronika dan mengambil topik Tugas Akhir di Laboratorium Elektronika Industri.

E-mail : roby.adiwibowo@gmail.com